TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM VALIDASI KEHADIRAN PERKULIAHAN DENGAN METODE K – NEAREST NEIGHBOR BERBASIS APLIKASI PERANGKAT BERGERAK

ADIAN LATIFA NURROHMAN
NRP 5113100031

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Dwi Sunaryono, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM VALIDASI KEHADIRAN PERKULIAHAN DENGAN METODE K – NEAREST NEIGHBOR BERBASIS APLIKASI PERANGKAT BERGERAK

ADIAN LATIFA NURROHMAN
NRP 5113100031

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Dwi Sunaryono, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

VALIDATION SYSTEM DESIGN OF COURSE ATTENDANCE WITH K – NEAREST NEIGHBOR BASED ON MOBILE APPLICATIONS

ADIAN LATIFA NURROHMAN
NRP 5113100031

Supervisor I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Supervisor II
Dwi Sunaryono, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM VALIDASI KEHADIRAN PERKULIAHAN DENGAN METODE K – NEAREST NEIGHBOR BERBASIS APLIKASI PERANGKAT BERGERAK

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
ADIAN LATIFA NURROHMAN
NRP : 5113 100 031

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Henning Titi Ciptaningtyas, S.Kom., M.Kom. .....
   NIP: 19840708 201012 2 003 (Pembimbing 1)

2. Dwi Sunaryono, S.Kom., M.Kom. .....
   NIP: 19720528 199702 1 001 (Pembimbing 2)

SURABAYA
MEI, 2017

v

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM VALIDASI KEHADIRAN PERKULIAHAN DENGAN METODE K – NEAREST NEIGHBOR BERBASIS APLIKASI PERANGKAT BERGERAK

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
ADIAN LATIFA NURROHMAN
NRP : 5113 100 031

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Henning Titi Ciptaningtyas, S.Kom., M.Kom.....
   NIP: 19840708 201012 2 004 (Pembimbing 1)

2. Dwi Sunaryono, S.Kom., M.Kom. .....
   NIP: 19720528 199702 1 001 (Pembimbing 2)

SURABAYA
MEI, 2017

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN SISTEM VALIDASI KEHADIRAN PERKULIAHAN DENGAN METODE K – NEAREST NEIGHBOR BERBASIS APLIKASI PERANGKAT BERGERAK

Nama Mahasiswa	**: Adian Latifa Nurrohman**
NRP	**: 5113100031**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Henning Titi Ciptaningtyas, S.Kom.,M.Kom.**
Dosen Pembimbing 2	**: Dwi Sunaryono, S.Kom., M.Kom.**

Abstrak

Perkembangan teknologi informasi di dunia semakin cepat, khususnya pada teknologi perangkat bergerak dan internet. Akses dan pengambilan informasi dikatakan semakin mudah dan cepat diakses melalui penggunaan perangkat bergerak dan *web*. Perkembangan ini juga mempengaruhi data sebagai validasi menggantikan kode sandi (*password*). Validasi merupakan sebuah proses yang wajib ada dimana diperlukan sebuah kebenaran data atau informasi, salah satunya adalah proses validasi kehadiran. Proses validasi sudah banyak diterapkan seiring dengan perkembangan teknologi informasi misalnya validasi kehadiran dengan menggunakan alat yang menerima inputan metode *fingerprin*t, validasi kehadiran menggunakan deteksi suara, dan lain-lain. Namun kelemahan dari penerapan yang sudah ada adalah kurangnya fleksibilitas. Fleksibilitas yang dimaksud adalah teknologi yang digunakan tidak bisa berpindah tempat dan diakses kapanpun. Oleh karena itu, tugas akhir ini akan mengimplementasikan sebuah aplikasi *mobile* validasi kehadiran berbasis android.

Aplikasi validasi kehadiran yang telah dibuat sudah dibatasi ruang lingkup penggunaannya. Pengguna dari aplikasi ini adalah mahasiswa Teknik Informatika ITS. Proses validasi

kehadiran pada aplikasi ini mengimplementasikan metode pencocokan lokasi mahasiswa terhadap kelas matakuliah berlangsung dan *QR Code* dari masing-masing kelas. Selanjutnya, lokasi yang didapatkan dari mahasiswa akan diklasifikan dengan menggunakan algoritma K-Nearest Neighbors.

Selain dengan algoritma K-Nearest Neighbors, data lokasi mahasiswa juga akan diklasifikasikan dengan metode rata-rata dan metode regresi linier. Prinsip kerja dari aplikasi ini akan membandingkan metode KNN dengan kedua metode tersebut. Kemudian, ditemukan kesamaan hasil dari ketiga metode. Apabila ketiga metode mendapatkan hasil yang sama, maka dapat dipastikan lokasi mahasiswa dengan tingkat akurasi yang tinggi.

Hasil dari pencarian lokasi mahasiswa, akan dicocokan dengan ruang kuliah pada jadwal mahasiswa yang terdapat dalam *database*. Jika terdapat kesamaan, maka mahasiswa berhak melakukan proses validasi kehadiran melalui *face recognition* atau fitur pencocokan tanda tangan yang terdapat pada aplikasi.

Kata kunci: Absensi, Validasi Kehadiran, Lokasi, KNN, Aplikasi Perangkat Bergerak

VALIDATION SYSTEM DESIGN OF COURSE ATTENDANCE WITH K – NEAREST NEIGHBOR BASED ON MOBILE APPLICATIONS

Student Name	**: Adian Latifa Nurrohman**
NRP	**: 5113100031**
Department	**: Teknik Informatika FTIF-ITS**
Supervisor 1	**: Henning Titi Ciptaningtyas, S.Kom.,M.Kom.**
Supervisor 2	**: Dwi Sunaryono, S.Kom., M.Kom.**

Abstract

The development of information technology in the world going faster, especially on mobile devices and internet technology. Access and retrieval of information is said to be increasingly easy and quickly accessible through the use of mobile and web devices. The development of this technology will replace the manual system that has been used because it is considered wasteful (resources) and not efficiency in the execution time. This development also affects the data as validation replaces the password (password). Validation is a mandatory process where there is a truth of data / information, one of which is the process of presence validation. Validation process has been widely applied in line with the development of information technology such as attendance validation using tools that accept input method fingerprint, attendance validation using voice detection, and others. But the disadvantage of the existing application is the lack of flexibility of this application. Flexibility in question is the technology used can not be moved and accessed anytime. Therefore, this final task will implement a mobile application based on android presence validation.

Attendance validation applications that have been created are limited in scope of use. Users of this application are

ITS Informatics Engineering students. The attendance validation process in this application implements the student location matching method against the course class and QR Code for all of the course class. Furthermore, the location obtained from the students will be classified using the K-Nearest Neighbors algorithm.

In addition to the K-Nearest Neighbors algorithm, student location data will also be classified by the Simple Average method and Linear Regression method. The working principle of this application will compare the KNN method with both methods. Later, found the similarity of the results of the three methods. If the three methods get the same results, then we can be sure the location of students with a high degree of accuracy.

Results from the student location search will be matched with the lecture room on the student schedule contained in the database. If there are similarities, then the student is entitled to perform the process of attendance validation through face recognition or signature matching featured in the application.

Keywords: Time Attendance, Attendance Validation, Location, QR Code, KNN, Simple Average, Linear Regression, Mobile Applications

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, karena dengan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"RANCANG BANGUN SISTEM VALIDASI KEHADIRAN PERKULIAHAN DENGAN METODE K – NEAREST NEIGHBOR BERBASIS APLIKASI PERANGKAT BERGERAK".** Dengan pengerjaan Tugas Akhir ini, penulis mencurahkan segala hal yang didapatkan selama kuliah di Teknik Informatika ITS dan dapat mengembangkannya menjadi sesuatu yang berguna.

Penulis dapat menyelesaikan Tugas Akhir ini tidak luput karena bantuan dari beberapa pihak. Penulis ingin menyampaikan terima kasih kepada:

1. Keluarga khususnya kedua orang tua yang telah memberikan semangat, dan senantiasa mengirimkan doa untuk kesuksesan penulis.
2. Ibu Henning Titi Ciptaningtyas, S.Kom.,M.Kom. selaku pembimbing I yang telah membimbing, dan memotivasi untuk penulis dalam menyelesaikan Tugas Akhir ini.
3. Bapak Dwi Sunaryono, S.Kom., M.Kom. selaku pembimbing II yang juga telah membimbing, dan memotivasi untuk penulis dalam mengerjakan Tugas Akhir ini.
4. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS.
5. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Teman-teman angkatan 2013 yang yang telah berbagi ilmu, dan memberi motivasi untuk penulis.
7. Semua pihak yang tidak bisa penulis sebutkan satu persatu.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Penulis berharap mendapatkan banyak kritik, saran, ataupun pengembangan Tugas Akhir ini pada Tugas Akhir selanjutnya.

Surabaya, Mei 2017

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR GRAFIK

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

DAFTAR PSEUDOCODE

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi di dunia semakin cepat, khususnya pada teknologi perangkat bergerak dan internet. Akses dan pengambilan informasi dikatakan semakin mudah dan cepat diakses melalui penggunaan perangkat bergerak dan *web*. Perkembangan ini juga mempengaruhi data sebagai validasi menggantikan kode sandi (*password*). Validasi merupakan sebuah proses yang wajib ada dimana diperlukan sebuah kebenaran data atau informasi, salah satunya adalah proses validasi kehadiran. Proses validasi sudah banyak diterapkan seiring dengan perkembangan teknologi informasi misalnya validasi kehadiran dengan menggunakan alat yang menerima inputan metode *fingerprint*, validasi kehadiran menggunakan deteksi suara, dan lain-lain. Namun kelemahan dari penerapan yang sudah ada adalah kurangnya fleksibilitas dari penerapan ini. Fleksibilitas yang dimaksud adalah teknologi yang digunakan tidak bisa berpindah tempat dan diakses kapanpun. Sehingga, untuk memenuhi dan menghilangkan kelemahan tersebut, maka dibuatlah sistem validasi kehadiran mahasiswa berbasis perangkat bergerak.

Sistem validasi kehadiran mahasiswa merupakan proses pencatatan kehadiran mahasiswa di kelas. Dengan adanya validasi kehadiran maka dapat membuktikan bahwa mahasiswa tersebut telah berada di dalam kelas. Validasi kehadiran dapat dilakukan dengan berbagai cara. Di Teknik Informatika ITS, sistem validasi kehadiran mahasiswa dilakukan secara manual, yaitu melakukan tanda tangan di kertas validasi kehadiran kelas. Hal ini menimbulkan celah kelemahan yaitu terjadi kecurangan dalam validasi kehadiran atau disebut titip absen. Titip absen merupakan kejadian dimana mahasiswa meminta teman satu kelas untuk menandatangani absen pada matakuliah tertentu, sehingga mahasiswa tersebut tidak perlu datang untuk menghadiri kelas.

Oleh karena itu, diperlukan suatu sistem yang dapat digunakan untuk mengatasi kecurangan dalam validasi kehadiran.

Aplikasi validasi kehadiran yang sudah dibuat telah dibatasi ruang lingkup penggunanya. Pengguna dari aplikasi ini adalah mahasiswa Teknik Informatika ITS. Proses validasi kehadiran pada aplikasi ini mengimplementasikan metode pencocokan lokasi mahasiswa terhadap kelas matakuliah berlangsung. Selanjutnya, lokasi yang didapatkan dari mahasiswa akan diklasifikasikan dengan menggunakan algoritma K-Nearest Neighbors, Simple Average, dan Regresi Linier. Ketiga metode ini akan dibandingkan dan dicari kesamaan hasilnya.

Hasil yang diharapkan dari pengerjaan tugas akhir ini berupa aplikasi perangkat bergerak yang menggantikan proses validasi kehadiran mahasiswa dari sistem manual melalui tanda tangan pada kertas presensi kehadiran kelas menjadi sistem *online* dengan metode pencocokan lokasi mahasiswa pada aplikasi *mobile*. Selain itu, dengan aplikasi ini diharapkan juga dapat memberikan kebenaran data absen (*data validation*) serta mempermudah pemantauan kehadiran mahasiswa di kelas untuk orang tua atau wali mahasiswa dan juga dosen wali di jurusan Teknik informatika ITS.

1.1 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka dapat dirumuskan beberapa permasalahan Tugas Akhir ini, antara lain:

1. Bagaimana cara merancang bangun aplikasi android sistem validasi kehadiran mahasiswa melalui pencocokan lokasi?
2. Bagaimana cara menggunakan Mobile Vision API untuk memindai *QR Code*?
3. Bagaimana cara penggunaan Google Maps API untuk mengoptimalkan GPS dalam pencarian lokasi?
4. Bagaimana cara menerapkan KNN pada aplikasi perangkat bergerak yang mencocokan posisi mahasiswa di kelas untuk validasi kehadiran?

5. Bagaimana cara menerapkan metode Rata-Rata (*Simple Average*) dan metode Regresi Linier untuk membandingkan hasil?

1.2 Batasan Masalah

Permasalahan pada Tugas Akhir ini memiliki beberapa batasan, diantaranya adalah sebagai berikut:

1. Sistem perangkat lunak dibangun untuk perangkat *mobile* dengan sistem operasi Android.
2. Sistem perangkat lunak dibangun menggunakan bahasa pemrograman Java, XML, dan dengan IDE Android Studio.
3. Masukkan perangkat lunak berupa data lokasi mahasiswa yang akan dicocokan dengan lokasi kelas mahasiswa tersebut akan kuliah.
4. Hasil yang akan diperoleh berupa notifikasi keberhasilan absen mahasiswa yang nantinya akan dikirimkan menuju *database*.
5. Proses validasi kehadiran akan berjalan ketika berada pada lokasi yang sudah ditentukan untuk melakukan validasi kehadiran, yaitu gedung Teknik Informatika ITS.
6. Proses validasi kehadiran akan berjalan ketika terhubung dengan jaringan internet ITS.

1.3 Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah:

1. Mampu merancang bangun aplikasi android sistem validasi kehadiran mahasiswa melalui pencocokan lokasi.
2. Mampu menerapkan Google Maps API untuk mengoptimalkan GPS.
3. Mampu menerapkan Mobile Vision API untuk memindai *QR Code*.
4. Membangun aplikasi validasi kehadiran berbasis lokasi dengan optimasi oleh KNN.

5. Mampu membandingkan metode KNN dengan metode Rata-rata, dan metode Regresi Linier.

1.4 Manfaat

Hasil dari pengerjaan Tugas Akhir ini memiliki manfaat untuk memperbaharui sistem validasi kehadiran mahasiswa dari sistem manual melalui tanda tangan pada absen kelas menjadi sistem *online* dengan menggunakan metode pencocokan lokasi aplikasi *mobile*. Data yang akan diperoleh merupakan data yang benar, lebih jelas, dan terperinci. Sehingga, pemantauan orangtua wali terhadap anaknya maupun dosen terhadap mahasiswanya akan lebih mudah.

1.5 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Proposal Tugas Akhir ini berisi tentang perencanaan "Rancang Bangun Sistem Validasi Kehadiran Perkuliahan Dengan Metode K - Nearest Neighbor Berbasis Aplikasi Perangkat Bergerak".

   Proposal Tugas Akhir ini terdiri dari deskripsi pendahuluan yang menjabarkan latar belakangan dan rumusan masalah yang mendasari dibangunnya aplikasi ini, batasan masalah dalam pembangunan aplikasi ini, serta tujuan dan manfaat yang diharapkan dapat dicapai dengan dibangunnya aplikasi ini. Selain itu, pada proposal Tugas Akhir ini juga terdapat tinjauan pusataka yang menjelaskan teori-teori yang menjadi dasar pembuatan tugas akhir ini, Ringkasan isi tugas akhir yang menggambarkan secara umum aplikasi yang akan dibangun dan algoritma yang digunakan, serta bagian metodologi dari penyusunan proposal tugas akhir ini.

2. Studi literatur

Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini adalah mengenai implementasi penggunaan Google Maps API serta algoritma KNN yang diperlukan untuk perancangan metode pencocokan lokasi pada perangkat *mobile* android. Studi literatur diambil dari buku, internet, maupun materi mata kuliah yang berhubungan dengan metode yang digunakan.

3. Analisis dan perancangan perangkat lunak

Analisis kebutuhan dan perancangan sistem dilakukan untuk merumuskan solusi yang tepat dalam pembuatan aplikasi serta kemungkinan yang dapat dilakukan untuk mengimplementasikan rancangan tersebut. Tahap desain meliputi arsitektur perangkat lunak yang digunakan, desain kelas-kelas yang terlibat dalam aplikasi, desain antarmuka, serta diagram-diagram yang mendukung pendeksripsian sistem aplikasi.

4. Implementasi perangkat lunak

Pembembangan aplikasi dalam Tugas Akhir akan menggunakan bahasa pemrograman Java, XML dengan kakas bantu IDE Android Studio. Selain itu, digunakan juga Google Maps API versi 3.27 dan metode K-Nearest Neighbor (KNN) untuk pembangunan metode pencocokan lokasi pada perangkat *mobile* android.

Pada tugas akhir ini menggunakan metode KNN karena metode ini dianggap dapat memproses data dengan cepat. Data-data lokasi yang didapatkan dapat segera dikelompokan dengan cepat dan memberikan efektifitas yang tinggi.

Untuk mengetahui kelemahan dan kelebihan metode KNN digunakan juga metode lain yaitu metode rata-rata (*Simple Average*) dan metode regresi linier. Kedua metode ini digunakan untuk membandingkan setiap metode dengan metode KNN.

Metode rata-rata (*Simple Average*) dipilih sebagai pembanding karena metode rata-rata lebih populer dan lebih mudah digunakan, dalam penghitungannya selalu mempertimbangkan semua nilai data, tidak peka terhadap penambahan jumlah data, variasinya paling stabil dan cocok digunakan untuk data yang homogen.

Metode regresi linier dipilih sebagai pembanding karena metode ini mampu mendeskripsikan fenomena data melalui terbentuknya suatu model hubungan yang bersifat numerik.

5. Pengujian dan evaluasi

   Pengujian dan evaluasi aplikasi perangkat lunak hasil dari Tugas Akhir ini akan diujicobakan pada mahasiswa Teknik Informatika ITS yang memiliki smartphone dengan GPS, sensor gerak, dan sensor cahaya. Sebelum pengujian, aplikasi validasi kehadiran akan di-install pada smartphone serta pengumpulan data lokasi kelas - kelas yang ada pada Teknik Informatika ITS sebagai data sample untuk *database* validasi kehadiran mahasiswa.

6. Penyusunan buku Tugas Akhir

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.6 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi latar belakang, tujuan, dan manfaat, perumusan masalah, batasan masalah, metodologi yang

digunakan, dan sistematika penulisan dari Tugas Akhir ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan mengenai dasar-dasar penunjang dan teori-teori yang digunakan dalam proses pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Perangkat Lunak

Bab ini berisi tentang algoritma dan desain dari sistem yang akan diimplementasikan pada Tugas Akhir ini.

Bab IV Implementasi

Bab ini berisi tentang implementasi dari rancangan yang telah dibuat pada Tugas Akhir ini.

Bab V Uji Coba dan Evaluasi

Bab ini menjelaskan mengenai kemampuan aplikasi dengan melakukan uji kebenaran dan uji kinerja sesuai dengan data yang diujikan dan banyaknya iterasi.

Bab VI Kesimpulan dan Saran

Bab ini membahas kesimpulan dari hasil uji coba yang telah dilakukan dan berisi tentang saran pengembangan.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Aplikasi Perangkat Bergerak (*Mobile Applications*)

Aplikasi perangkat bergerak atau *mobile applications* adalah pengembangan aplikasi untuk perangkat genggam seperti PDA, asisten digital perusahaan atau telepon genggam, *smartphone* dan sebagainya. Aplikasi ini sudah ada pada telepon selama manufaktur, atau didownload oleh pelanggan dari toko aplikasi dan dari distribusi perangkat lunak *mobile* platform yang lain[1].

Platform smartphone terpopuler yang mendukung aplikasi perangkat bergerak saat ini adalah Android, iOS, Windows Phone dan BlackBerry[2].

2.2 Android

Android adalah software untuk perangkat *mobile* yang mencakup sistem operasi, middleware dan aplikasi kunci. Pengembangan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Serangkaian aplikasi inti Android antara lain klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain.

Dengan menyediakan sebuah platform pengembangan yang terbuka, pengembang Android menawarkan kemampuan untuk membangun aplikasi yang sangat kaya dan inovatif. Pengembang bebas untuk mengambil keuntungan dari perangkat keras, akses

informasi lokasi, menjalankan background services, mengatur alarm, tambahkan pemberitahuan ke status bar, dan banyak lagi.

Android bergantung pada versi Linux 2.6 untuk layanan sistem inti seperti keamanan, manajemen memori, manajemen proses, network stack, dan model driver. Kernel juga bertindak sebagai lapisan abstraksi antara hardware dan seluruh software stack[3].

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode–kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Services (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD)[3].

2.3 *Global Positioning System* (GPS)

GPS (*Global Positioning System*) atau sistem peletakkan adalah sistem navigasi yang berbasiskan satelit yang saling berhubungan yang berada di orbitnya. Satelit-satelit itu milik Departemen Pertahanan (*Department of Defense*) Amerika Serikat

yang pertama kali diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24 satelit[4].

Global positioning system (GPS) merupakan operasi menyeluruh pertama dari sistem satnav, dan di desain, serta dioperasikan untuk militer. GPS menggunakan notifikasi, yang disebut *Notice Advisories to Navstar Users* (NANUs), untuk menginformasikan pengguna tentang perubahan konstelasi GPS. Kemudian, NANUs memberikan karakteristik sinyal GPS yang mendetail. Perkembangan GPS pada tahun 1970an dibangun berdasarkan konsep teknis yang sebelumnya dikembangkan dan dieksplorasi, seperti navigasi radio dari pemancar terrestrial[5].

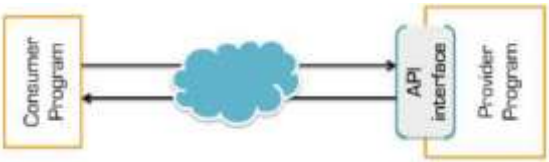
Pada tahun 1980-an GPS digunakan untuk kepentingan sipil. GPS dapat digunakan dimanapun juga dalam 24 jam. Posisi unit GPS akan ditentukan berdasarkan titik-titik koordinat derajat lintang dan bujur[4].

2.4 API

API atau *Application Programming Interface* adalah sekumpulan perintah, fungsi, serta protocol yang dapat digunakan oleh programmer saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi. API juga dapat menjelaskan cara sebuah tugas tertentu dilakukan. Sebuah API juga dapat digunakan untuk menspesifikasikan cara komponen aplikasi saling berinteraksi. Dengan bahasa yang lebih sederhana, API adalah fungsi-fungsi pemrograman yang disediakan oleh aplikasi atau layanan agar layananan tersebut bisa di integrasikan dengan aplikasi yang dibuat. Di dalam pemrograman android, API biasanya digunakan untuk mempermudah pertukaran data dari atau ke dalam *server*.

API merupakan *software interface* yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk library dan menjelaskan bagaimana agar suatu software dapat berinteraksi dengan software lain. Penjelasan ini dapat dicontohkan dengan analogi apabila akan dibangun suatu rumah. Dengan menyewa kontraktor yang dapat menangani bagian yang berbeda, pemilik rumah dapat memberikan tugas yang perlu dilakukan oleh kontraktor tanpa harus mengetahui bagaimana cara kontraktor menyelesaikan pekerjaan tersebut. Dari analogi tersebut, rumah merupakan software yang akan dibuat, dan kontraktor merupakan API yang mengerjakan bagian tertentu dari software tersebut tanpa harus diketahui bagaimana prosedur dalam melakukan pekerjaan tersebut[6].

Interface pada *software* merupakan suatu *entry points* yang digunakan untuk mengakses seluruh *resources* yang terdapat di dalam software tersebut. Dengan adanya API, maka terdapat aturan bagaimana software dapat berinteraksi dengan software lain untuk mengakses *resources* melalui interface yang telah tersedia.



Gambar 2.1 Skema Konektivitas API Antar Software

Secara struktural, API merupakan spesifikasi dari suatu data structure, objects, functions, beserta parameter-parameter yang diperlukan untuk mengakses resource dari aplikasi tersebut. Seluruh spesifikasi tersebut membentuk suatu interface yang dimiliki oleh aplikasi untuk berkomunikasi dengan aplikasi lain, dan API dapat digunakan dengan berbagai bahasa programming, ataupun hanya dengan menggunakan URL (*Uniform Resource Locator*) yang telah disediakan oleh suatu *web*site. API dapat diklasifikasikan menjadi beberapa kategori, hal ini dilihat dari

abstraksi apa yang dideskripsikan di dalam sistem. Kategori-kategori ini diantaranya:

2.5 *Quick Response Code (QR Code)*

Kode QR (Quick Response) merupakan bentuk evaluasi dari *barcode* yang biasanya kita lihat pada sebuah produk. Kode Qr berbentuk jajaran persegi berwarna hitam berbentuk seperti *barcode* tetapi dengan tampilan lebih ringkas[7].

Gambar 2.2 Contoh QR Code

Bar code adalah metode pengumpulan data yang cepat, mudah, akurat dan otomatis. *Barcode* memungkinkan produk dilacak dengan efisien dan akurat menggunakan sistem entri data manual. *Barcode scanner* hanya digunakan untuk mengenali *barcode*, dan harga *barcode scanner* itu mahal. Kini, ponsel bisa menerapkan banyak aplikasi baru, seperti pengambilan foto dan pemotretan film dengan menggunakan kamera yang tersedia. Jadi, pendekatan yang menarik adalah melakukan *scan barcode* dengan kamera mereka dan menguraikannya dengan perangkat lunak yang berjalan di ponsel[8].

2.6 *K - Nearest Neighbors (KNN)*

Algoritma K-Nearest Neighbor (KNN) merupakan sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek

tersebut. KNN termasuk algoritma *supervised learning* dimana hasil dari *query instance* yang baru diklasifikan berdasarkan mayoritas dari kategori pada KNN. Kelas yang paling banyak muncul itu yang akan menjadi kelas hasil klasifikasi. Tujuan dari algoritma ini adalah mengklasifikasikan objek baru berdasarkan atribut dan *training sample*[9].

Algoritma K- Nearest Neighbor menggunakan klasifikasi ketetanggaan (*neighbor*) sebagai nilai prediksi dari *query instance* yang baru. Algoritma ini sederhana, bekerja berdasarkan jarak terpendek dari *query instance* ke *training sample* untuk menentukan ketetanggaannya[10].

Teknik *K-Nearest Neighbor* dengan melakukan langkah-langkah yaitu mulai input: Data *training*, label data *training*, *k*, data *testing*[11]:

a. Untuk semua data *testing*, hitung jaraknya ke setiap data training
b. Tentukan k data *training* yang jaraknya paling dekat dengan data
c. *Testing*
d. Periksa label dari k data ini
e. Tentukan label yang frekuensinya paling banyak
f. Masukan data *testing* ke kelas dengan frekuensi paling banyak
g. Berhenti

Untuk menghitung jarak antara dua titik x dan y bisa digunakan jarak Euclidean pada persamaan 2.1 berikut ini:

$$d(X_1, Y_2) = \sum_{l} \left| \frac{n_{1i}}{n1} - \frac{n_{2i}}{n_2} \right| \qquad (2.1)$$

Yang mana X1, l = 1, 2, adalah atribut kategori, dan n1i, n1 mewakili frekuensi yang sesuai.

Kelebihan dari Algoritma K-Nearest Neighbor: [12]

- *Robust* terhadap data yang *noisy*
- Efektif jika training data berjumlah banyak

2.7 Metode Rata-Rata atau Simple Average (SA)

Dasar pemikiran metode ini, yaitu menghitung nilai tengah untuk setiap waktu dengan cara menjumlahkan seluruh nilai observasi sebelumnya dibagi jumlah data. Rumus dari metode ini yaitu pada persamaan 2.2 berikut ini[13]:

$$F_n = \frac{X_1 + X_2 + X_3 + ... + X_n}{n} \quad (2.2)$$

Kelebihan metode rata-rata[14]:

1. Rata-rata lebih populer dan lebih mudah digunakan.
2. Dalam satu set data, rata-rata selalu ada dan hanya ada satu rata-rata.
3. Dalam penghitungannya selalu mempertimbangkan semua nilai data.
4. Tidak peka terhadap penambahan jumlah data.
5. Variasinya paling stabil.
6. Cocok digunakan untuk data yang homogen.

Kelemahan metode rata-rata[14]:

1. Sangat peka terhadap data ekstrim. Jika data ekstrimnya banyak, rata-rata menjadi kurang mewakili (representatif).
2. Tidak dapat digunakan untuk data kualitatif.
3. Tidak cocok untuk data heterogen.

2.8 Metode Regresi Linier

Regresi linier adalah metode statistika yang digunakan untuk membentuk model hubungan antara variabel terikat (dependen; respon; Y) dengan satu atau lebih variabel bebas (independen, prediktor, X). Apabila banyaknya variabel bebas hanya ada satu, disebut sebagai regresi linier sederhana, sedangkan apabila terdapat lebih dari 1 variabel bebas, disebut sebagai regresi linier berganda.

Regresi linier sederhana adalah regresi yang melibatkan hubungan antara satu variabel tak bebas (Y) dihubungan dengan satu variabel bebas (X). Bentuk umum persamaan regresi linier sederhana yaitu pada persamaan 2.3 berikut ini[15]:

$$Y = a + bX \tag{2.3}$$

Dimana:
y = variabel tak bebas
a = intersep (titik potong kurva terhadap sumbu y)
b = kemiringan (*slope*) kurva linear
x = variabel bebas

Analisis regresi setidak-tidaknya memiliki 3 kegunaan, yaitu
1. Untuk tujuan deskripsi dari fenomena data atau kasus yang sedang diteliti,
2. Untuk tujuan kontrol, serta
3. Untuk tujuan prediksi.

Regresi mampu mendeskripsikan fenomena data melalui terbentuknya suatu model hubungan yang bersifatnya numerik. Regresi juga dapat digunakan untuk melakukan pengendalian (kontrol) terhadap suatu kasus atau hal-hal yang sedang diamati melalui penggunaan model regresi yang diperoleh. Selain itu, model regresi juga dapat dimanfaatkan untuk melakukan prediksi untuk variabel terikat. Namun kekurangan dari metode regresi linier adalah data-data yang diukur harus linier untuk memperoleh hasil yang baik[16].

Data untuk variabel independen X pada regresi linier bisa merupakan data pengamatan yang tidak ditetapkan sebelumnya oleh peneliti (obsevational data) maupun data yang telah ditetapkan (dikontrol) oleh peneliti sebelumnya (experimental or fixed data). Perbedaannya adalah bahwa dengan menggunakan

fixed data, informasi yang diperoleh lebih kuat dalam menjelaskan hubungan sebab akibat antara variabel X dan variabel Y. Sedangkan, pada observational data, informasi yang diperoleh belum tentu merupakan hubungan sebab-akibat. Untuk fixed data, peneliti sebelumnya telah memiliki beberapa nilai variabel X yang ingin diteliti. Sedangkan, pada observational data, variabel X yang diamati bisa berapa saja, tergantung keadaan di lapangan. Biasanya, fixed data diperoleh dari percobaan laboratorium, dan observational data diperoleh dengan menggunakan kuesioner.

Dalam suatu model regresi kita akan menemukan koefisien-koefisien. Koefisien pada model regresi sebenarnya adalah nilai duga parameter di dalam model regresi untuk kondisi yang sebenarnya (true condition), sama halnya dengan statistik *Simple Average* (rata-rata) pada konsep statistika dasar. Hanya saja, koefisien-koefisien untuk model regresi merupakan suatu nilai rata-rata yang berpeluang terjadi pada variabel Y (variabel terikat) bila suatu nilai X (variabel bebas) diberikan. Koefisien regresi dapat dibedakan menjadi 2 macam, yaitu:

1. Intersep (intercept)

   Intersep, definisi secara metematis adalah suatu titik perpotongan antara suatu garis dengan sumbu Y pada diagram/sumbu kartesius saat nilai X = 0. Sedangkan definisi secara statistika adalah nilai rata-rata pada variabel Y apabila nilai pada variabel X bernilai 0. Dengan kata lain, apabila X tidak memberikan kontribusi, maka secara rata-rata, variabel Y akan bernilai sebesar intersep. Perlu diingat, intersep hanyalah suatu konstanta yang memungkinkan munculnya koefisien lain di dalam model regresi. Intersep tidak selalu dapat atau perlu untuk diinterpretasikan. Apabila data pengamatan pada variabel X tidak mencakup nilai 0 atau mendekati 0, maka intersep tidak memiliki makna yang berarti, sehingga tidak perlu diinterpretasikan.

2. Slope

   Secara matematis, *slope* merupakan ukuran kemiringan dari suatu garis. *Slope* adalah koefisien regresi untuk variabel X

(variabel bebas). Dalam konsep statistika, *slope* merupakan suatu nilai yang menunjukkan seberapa besar kontribusi (sumbangan) yang diberikan suatu variabel X terhadap variabel Y. Nilai slope dapat pula diartikan sebagai rata-rata pertambahan (atau pengurangan) yang terjadi pada variabel Y untuk setiap peningkatan satu satuan variabel X.

2.9 *Server*

Server adalah komputer yang dirancang untuk memproses permintaan dan mengirimkan data ke komputer lain (klien) melalui jaringan lokal atau internet. Meskipun komputer yang menjalankan perangkat lunak khusus dapat berfungsi sebagai *server*, penggunaan kata yang paling umum merujuk pada mesin bertenaga tinggi yang sangat besar yang berfungsi untuk mengirim dan mengambil data pada internet[17].

2.10 *Database*

Database, juga disebut *database* elektronik, kumpulan data, atau informasi, yang khusus disusun untuk pencarian dan pencarian cepat oleh komputer. *Database* disusun untuk memudahkan penyimpanan, pengambilan, modifikasi, dan penghapusan data bersamaan dengan berbagai operasi pengolahan data. Sistem manajemen basis data (DBMS) mengekstrak informasi dari *database* sebagai respon terhadap *query*.

Basis data adalah kumpulan data yang saling berhubungan secara logikal serta deskripsi dari data tersebut, yang dirancang untuk memenuhi kebutuhan informasi suatu organisasi. Basis Data adalah sebuah penyimpanan data yang besar yang bisa digunakan oleh banyak pengguna dan departemen. Semua data terintegrasi dengan jumlah duplikasi yang minimum. Basis data tidak lagi dipegang oleh satu departemen, tetapi dibagikan ke seluruh departemen pada perusahaan. Basis data itu sendiri tidak hanya memegang data operasional organisasi tetapi juga penggambaran dari data tersebut[18].

Basis data adalah kumpulan data store yang terintegrasi yang diatur dan di kontrol secara sentral. Sebuah basis data biasanya menyimpan ribuan class. Informasi yang disimpan termasuk class attribute dan relasi antar class. Basis data juga menyimpan informasi yang deksriptif seperti nama atribut, pemberian batasan suatu nilai, dan kontrol akses untuk data-data yang sensitif[19].

Dapat disimpulkan basis data adalah penyimpanan data yang terstruktur, terintegrasi dan saling berkaitan dengan elemen-elemen penghubungnya dan dapat di akses dengan berbagai cara, oleh karena itu basis data juga bisa didefinisikan sebagai kumpulan yang menggambarkan sendiri dari catatan yang terintegrasi dan penggambaran dari data dikenal sebagai sistem katalog (atau kamus data atau metadata). Definisi data disini dibedakan dari program aplikasi, yang umumnya sama dengan pendekatan pengembangan modern perangkat lunak, dimana definisi internal dan eksternal dari sebuah objek dipisahkan. Salah satu keuntungan dari pendekatan tersebut adalah abstraksi data dimana kita dapat mengubah definisi internal dari sebuah objek tanpa mempengaruhi pengguna dari objek jika definisi eksternal objek tersebut tidak berubah.

2.11 JSON

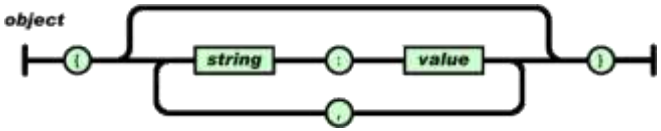
JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.
JSON terbuat dari dua struktur[20]:

- Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
- Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

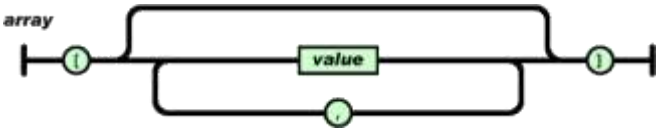
JSON menggunakan bentuk sebagai berikut[20]:

1. **Objek** adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan **{** (kurung kurawal buka) dan diakhiri dengan **}** (kurung kurawal tutup). Setiap nama diikuti dengan **:** (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh **,** (koma).



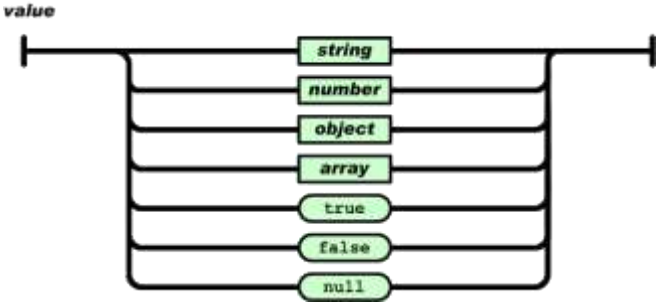
Gambar 2.3 Object

2. **Larik** adalah kumpulan nilai yang terurutkan. Larik dimulai dengan **[** (kurung kotak buka) dan diakhiri dengan **]** (kurung kotak tutup). Setiap nilai dipisahkan oleh **,** (koma).



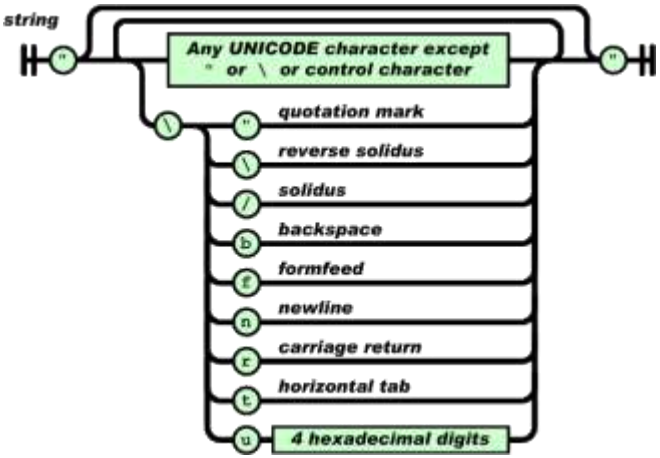
Gambar 2.4 Array/Larik

3. **Nilai** (*value*) dapat berupa sebuah **string** dalam tanda kutip ganda, atau *angka*, atau **`true`** atau **`false`** atau **`null`**, atau sebuah *objek* atau sebuah *larik*. Struktur-struktur tersebut dapat disusun bertingkat.



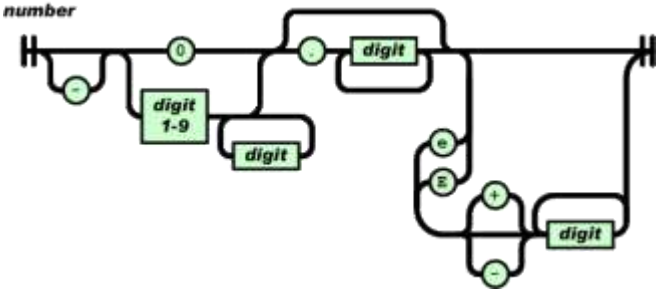
Gambar 2.5 Value (Nilai)

4. **String** adalah kumpulan dari nol atau lebih karakter Uni*code*, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.



Gambar 2.6 String

5. **Angka** adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Gambar 2.7 Number atau Angka

Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

BAB III
ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dijelaskan perancangan perangkat lunak yang dibuat. Perancangan akan dibagi menjadi dua tahapan utama, yaitu perancangan arsitektur program dan peracancangan alur proses utama program. Pada bab ini juga akan dijelaskan mengenai gambaran umum setiap proses utama program dalam diagram alir beserta penjelasannya.

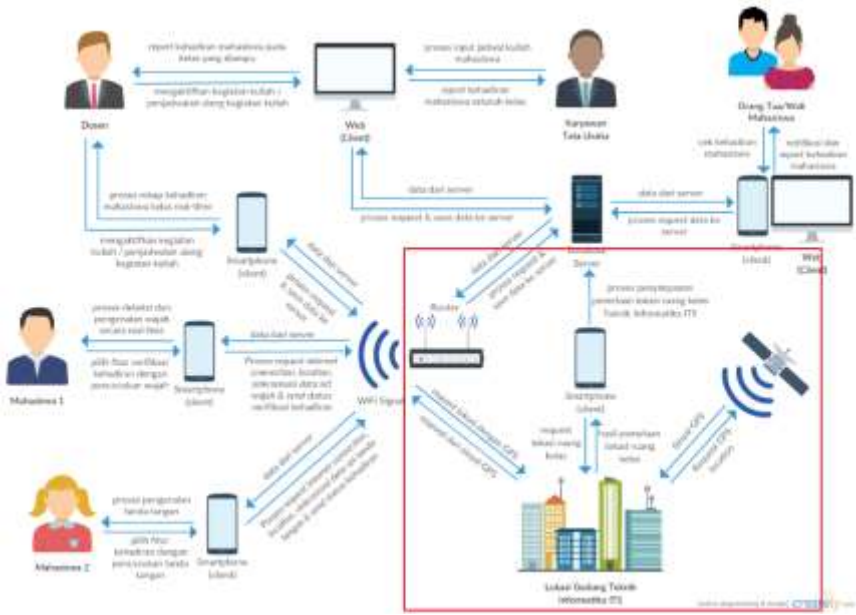
3.1 Penjelasan Umum Aplikasi

Gambar 3.1 merupakan diagram arsitektur sistem yang menunjukkan cara kerja keseluruhan sistem ini, mulai dari pemindaian *QR Code*, pencarian lokasi dari GPS *smartphone*, ekstraksi fitur, pengklasifikasian data, hingga pengiriman status validasi kehadiran ke *database server*.

Pada awalnya, mahasiswa harus melakukan proses login ke dalam aplikasi *mobile* validasi kehadiran mahasiswa. Pada saat proses login berjalan, aplikasi memastikan *smartphone* sudah terhubung ke jaringan *internet wi-fi* ITS. Setelah proses login berhasil, mahasiswa diharuskan untuk melakukan proses pemindaian *QR Code* dari kelas yang sesuai. Kemudian setelah proses pemindian selesai, aplikasi akan mengaktifkan fitur pencocokan lokasi yang memiliki dua tahap. Tahap pertama, aplikasi akan mengambil data lokasi dari mahasiswa saat kelas sudah dimulai yang akan dijadikan dataset untuk proses ekstraksi fitur. Tahap kedua, data tersebut akan diolah oleh algoritma KNN. Data yang diolah akan dibandingkan dengan data lokasi-lokasi kelas yang ada pada *database* lokal. Hasil klasifikasi akan disimpan sementara pada *database* lokal.

Dari kedua tahap tersebut akan diulangi sebanyak dua kali untuk proses pencocokan lokasi saat kelas berlangsung, dan saat kelas telah usai. Ketiga data klasifikasi tersebut akan menentukan

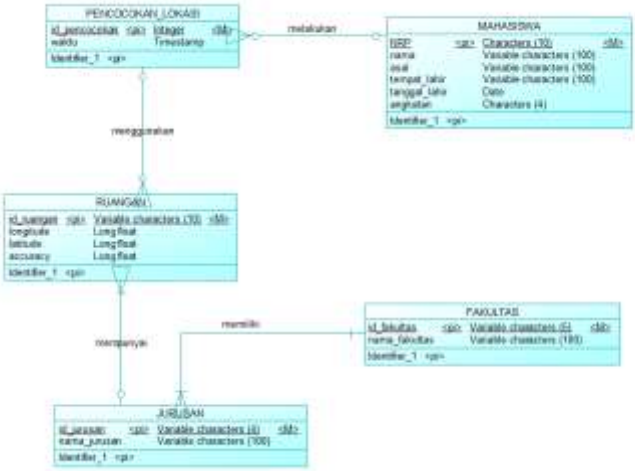
validasi kehadiran dari mahasiswa. Data validasi kehadiran akan disimpan pada *database* lokal dan akan disinkronisasikan dengan *database server*.



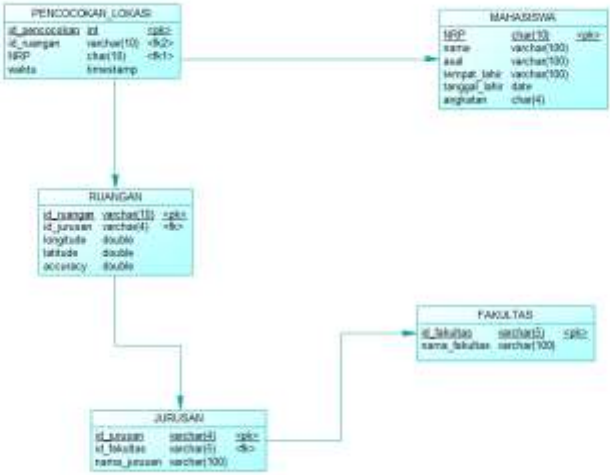
Gambar 3.1 Gambaran Umum Arsitektur Perangkat Lunak yang akan dibuat

3.2 Perancangan *Database* Aplikasi

Tahap perancangan *database* terdiri dari tabel fakultas, jurusan, ruangan, mahasiswa, dan pencocokan lokasi.



Gambar 3.2 Conseptual Data Model Aplikasi



Gambar 3.3 Physical Data Model Aplikasi

Setiap data yang berhasil diklasifikasikan dengan ketiga metode, akan menghasilkan kesimpulan metode yang memberikan hasil terbaik. Kesimpulan tersebut akan disimpan sebagai hasil pencocokan lokasi yang digunakan dalam validasi kehadiran.

3.3 Perancangan Alur Proses Aplikasi

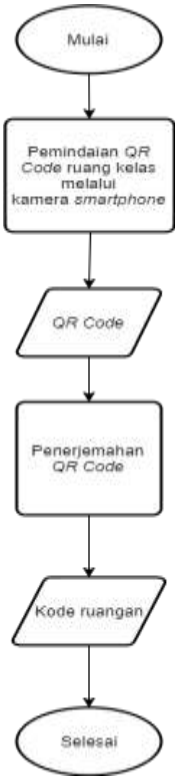
Tahap perancangan alur proses aplikasi, yaitu pemindaian *QR Code*, proses pengambilan data ruangan pada *database*, proses pengambilan data lokasi via GPS, proses penentuan lokasi mahasiswa, dan proses penyimpanan data validasi kehadiran.

3.3.1 Proses Pemindaian *QR Code*

Proses pemindaian *QR Code* merupakan proses mahasiswa melakukan pengambilan gambar *QR Code* yang telah disediakan pada masing-masing kelas. *QR Code* ini digunakan untuk mengetahui keberadaan mahasiswa agar nantinya mahasiswa dapat melakukan validasi kehadiran dengan fitur selanjutnya, yaitu tanda tangan atau *face recognition*. *String* yang berada pada *QR Code* merupakan hasil *encoding* dari kode ruang kelas dengan menggunakan Base64 atau MD5 dan memiliki pola berupa nama kelas dan *salt*, yaitu "Informatika Sikemas".

Gambar 3.4 QR Code dari IF-101 Informatika Sikemas dengan Base64

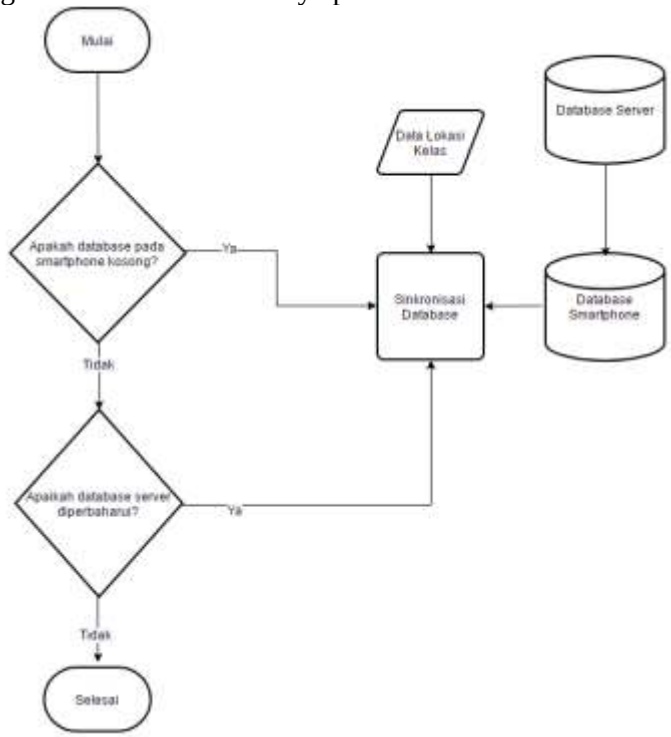
Gambar 3.5 QR Code IF-102 Informatika Sikemas dengan MD5



Gambar 3.6 Diagram alir Proses Pemindaian QR Code

3.3.2 Proses Pengambilan Data Ruangan Pada *Database*

Proses pengambilan data ruangan pada *database* adalah pengambilan data ruangan sebagai acuan untuk pengelompokan lokasi mahasiswa saat itu. Mahasiswa berada pada ruangan tertentu yang sudah dimiliki kriterianya pada *database*.



Gambar 3.7 Diagram alir Proses Pengambilan Data Ruangan Pada Database

Gambar 3.7 merupakan gambaran proses pengambilan data ruangan pada *database*. Data ruangan pada *database* lokal akan disinkronisasikan dengan *database* yang berada pada *server*. Data ruangan ini nantinya akan dicocokan dengan keberadaan mahasiswa saat itu. Dengan beberapa metode yang digunakan, akan dihasilkan kesimpulan tentang lokasi mahasiswa yang dekat dengan ruangan tertentu dan apakah mahasiswa tersebut dapat divalidasi kehadirannya.

3.3.3 Proses Pengambilan Data Lokasi via GPS

Proses pengambilan data lokasi via GPS merupakan proses pengambilan data lokasi mahasiswa saat itu. Mahasiswa akan diketahui lokasinya menggunakan GPS yang telah aktif pada *smartphone*. Apabila GPS belum diaktifkan, maka proses pengambilan data lokasi tidak akan dapat dilakukan.

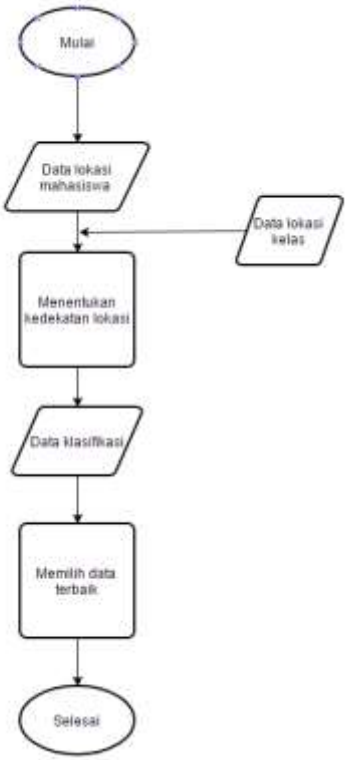
Selama proses berlangsung, akan dibantu dengan Google Maps API. yang memanfaatkan sinyal, dan beberapa sensor yang terdapat pada *smartphone*. Tingkat akurasi yang dihasilkan, lebih tinggi dibandingkan dengan penggunaan GPS tanpa Google Maps API. Jangkauan dari area atau titik lokasi mahasiswa berada, akan makin sempit yang membuat keberadaan mahasiswa pada GPS jadi lebih dekat dengan keberadaan mahasiswa sebenarnya.

Gambar 3.8 Diagram alir Proses Pengambilan Data Lokasi via GPS

Gambar 3.8 menggambarkan proses pengambilan data lokasi mahasiswa pada GPS. Data lokasi diambil dengan menggunakan *fused location* yang terdapat pada Google Maps API. Hasil dari data lokasi mahasiswa tersebut terdiri dari *longitude*, *latitude*, dan *accuracy*.

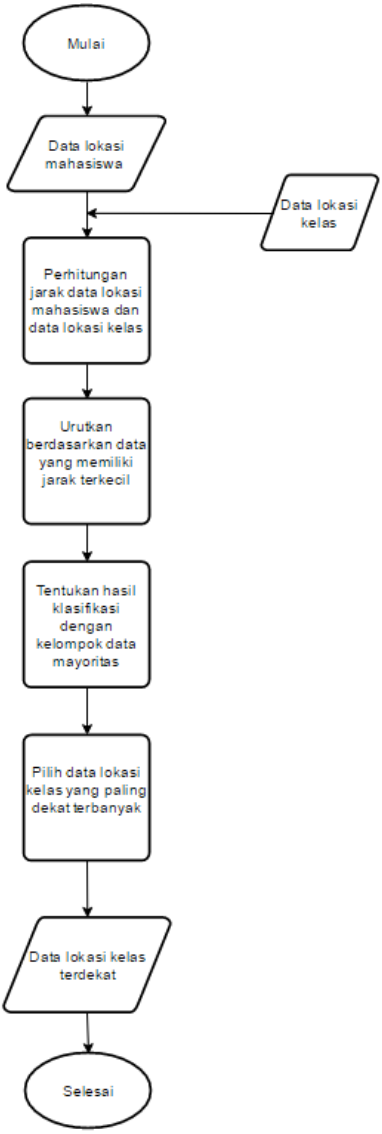
3.3.4 Proses Penentuan Lokasi Mahasiswa

Proses penentuan lokasi mahasiswa merupakan proses pengolahan data lokasi mahasiswa. Data diolah dengan 3 metode, yaitu KNN, *Simple Average*, dan Regresi Linier. Pengolahan data dengan ketiga metode tersebut berbeda-beda.



Gambar 3.9 Diagram alir proses penentuan lokasi mahasiswa

Gambar 3.9 menjelaskan alur keseluruhan saat menentukan lokasi mahasiswa. Mulai dari *input* data lokasi mahasiswa, metode yang digunakan, data lokasi kelas, menentukan kedekatan lokasi, dan memilih data yang terbaik.

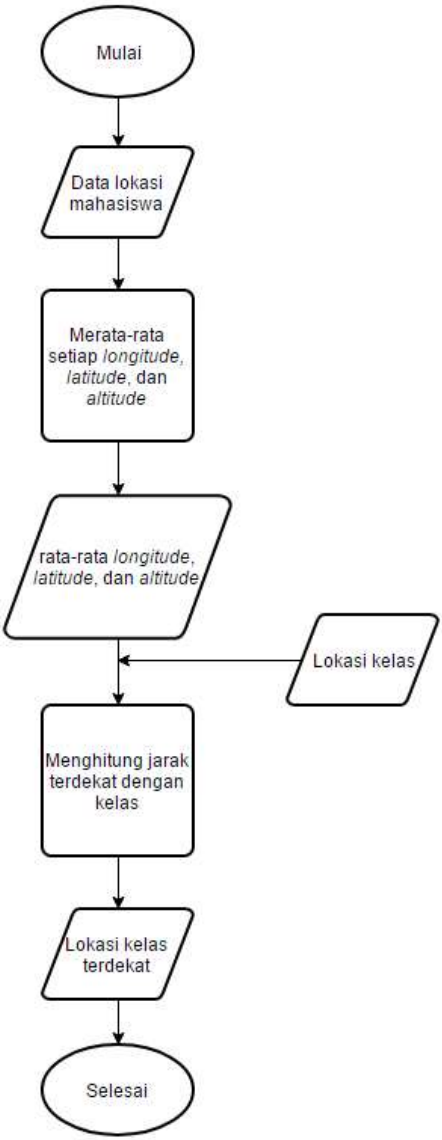


Gambar 3.10 Diagram alir metode KNN yang digunakan dalam menentukan lokasi mahasiswa

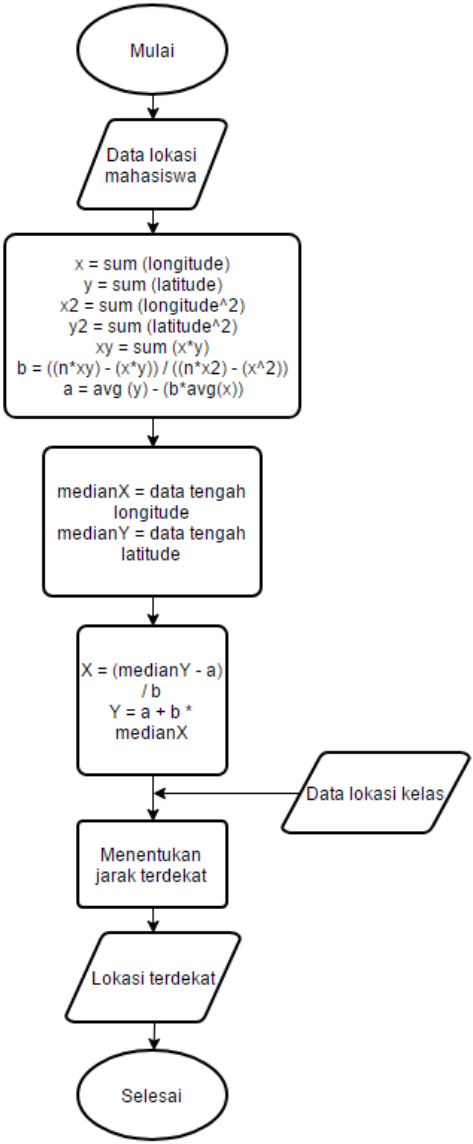
Metode KNN merupakan metode mengelompokan data dengan tetangga terdekat atau objek yang dianggap memiliki jarak terdekat dari *dataset* yang ada. Pada Gambar 3.10 menunjukkan proses kerja metode KNN dalam menentukan lokasi mahasiswa. Metode KNN membutuhkan data lokasi kelas dan lokasi mahasiswa saat itu. Dimulai dengan data lokasi mahasiswa saat itu yang akan dihitung jaraknya dengan setiap data lokasi kelas. Setelah jarak dari setiap data lokasi mahasiswa saat itu dan data lokasi kelas diketahui, proses selanjutnya adalah mengurutkan data yang memiliki jarak terkecil. Jarak terkecil ini berdasarkan pada *euclidean distance*. Kemudian data-data tersebut dikelompokan sesuai dengan banyaknya data lokasi kelas yang terpilih sebagai jarak terdekat, lalu ambil yang terbanyak. Data ini yang merupakan hasil dari metode KNN.

Metode rata-rata bekerja dengan mencari titik tengah dari setiap data yang ada. Dalam hal ini titik tengah antara masing-masing koordinat *longitude, latitude,* dan *altitude* yang dilakukan dengan menjumlahkan, lalu membagi dengan banyaknya data (n). Metode rata-rata mirip dengan mencari data tengah atau *median* dengan data diurutkan sebelumnya. Hasil dari data tengah cenderung memiliki angka penting yang lebih sedikit dibandingkan dengan metode rata-rata.

Pada Gambar 3.11 metode rata-rata diawali dengan *input* data lokasi mahasiswa. Data tersebut akan dicari rata-ratanya dengan dijumlahkan, lalu dibagi dengan banyaknya data atau iterasi. Data yang dicari rata-ratanya merupakan data *longitude*, *latitude*, dan *altitude*. Setelah mendapatkan rata-rata dari setiap data, akan dilakukan perhitungan jarak dengan lokasi kelas. Data lokasi kelas satu per satu akan dibandingkan, dan diurutkan untuk mendapatkan jarak terpendek yang merupakan hasil dari metode ini.



Gambar 3.11 Diagram alir metode rata-rata dalam menentukan lokasi mahasiswa



Gambar 3.12 Diagram alir metode regresi linier dalam menentukan lokasi mahasiswa

Metode regresi linier merupakan Teknik statistik untuk menginvestigasi dan memodelkan hubungan antar variabel. Regresi linier jika digambarkan sebagai grafik akan menghasilkan persamaan garis linier atau lurus yang mewakili nilai-nilai dari semua variabel.

Gambar 3.12, metode regresi linier diawali dengan jumlah dari tiap komponen data lokasi mahasiswa. Variabel x mewakili penjumlahan seluruh data *longitude.* Variabel y merupakan penjumlahan seluruh data *latitude.* Variabel x2 merupakan penjumlahan pangkat dua dari tiap data *longitude*. Variabel y2 merupakan penjumlahan pangkat dua dari tiap data *latitude.* Variabel xy merupakan penjumlahan dari perkalian *longitude* dan *latitude* tiap iterasi. Selanjutnya, menghitung koefisien a dan b dengan rumus:

$$\text{a} = \frac{y}{n} - (b)\frac{x}{n} \qquad \text{b} = \frac{(n)(xy) - (x)(y)}{(n)(x2) - (x^2)}$$

Kemudian mencari *median* dari x dan y, lalu dimasukan pada persamaan:

$$Y = a + bX$$

Persamaan tersebut akan menghasilkan *longitude* dan *latitude* baru yang disimbolkan oleh x dan y. Tahap selanjutnya, menentukan jarak data baru tersebut dengan data lokasi kelas. Tahap akhir dari proses ini adalah mendapatkan data lokasi kelas dengan jarak terpendek dengan mengurutkan data jarak sebelumnya. Data ini yang akan disimpan dan digunakan sebagai validasi kehadiran mahasiswa.

3.3.5 Proses Penyimpanan Data Validasi Kehadiran

Penentuan lokasi mahasiswa yang telah dilakukan dengan 3 metode pada pembahasan sebelumnya, menghasilkan beberapa data dengan jarak tertentu dari setiap lokasi kelas. Selanjutnya, data-data tersebut akan dicari kesamaannya, sehingga lokasi mahasiswa dapat disimpulkan dengan data lokasi kelas mayoritas. Setelah mendapatkan kesimpulan, data tersebut akan disimpan menuju *database server*.

Gambar 3.13 Diagram alir penyimpanan data validasi kehadiran

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Lingkungan yang akan digunakan untuk melakukan proses implementasi adalah Android Studio versi 2.2.6 dengan spesifikasi computer Intel Core i7 RAM 16gb yang bekerja pada sistem operasi *Windows 10 64-bit.*

4.2 Pemasangan *Library* Mobile Vision, Google API, dan OpenCSV

Pada bagian ini dijelaskan mengenai cara pemasangan *libray* pada *project* Android Studio, yaitu Google API, dan OpenCSV. Langkah-langkah pemasangan *library* Google API adalah sebagai berikut:

1. Klik menu Tools, lalu Android.
2. Cari dengan kata kunci “Google Services”.
3. Centang, lalu OK.
4. Tambahkan pada AndroidManifest.xml

```
1.  <meta-data
2.      android:name="com.google.android.gms.version"

3.      android:value="@integer/google_play_services_v
    ersion" />
```

Langkah-langkah pemasangan *library* openCSV adalah sebagai berikut:

1. Tahan dan lepas *file* opencsv-3.9.jar yang berhasil diunduh ke app/libs.
2. Tambahkan "compile files('libs/opencsv-3.9.jar')" pada app/build.gradle bagian *dependencies.*

4.3 Implementasi QR Code Scanner

Implementasi *QR Code Scanner* digunakan untuk mengaktifkan fitur lain, seperti *face recognition* dan pencocokan tanda tangan. Sebelum aplikasi melakukan proses pencarian lokasi mahasiswa, melakukan pemindaian *QR Code*. Hasil dari *QR Code* berupa *string* dengan Base64 *encode* atau MD5 dengan pola nama kelas dan *salt*, yaitu "Informatika Sikemas".

```
1  open camera
2  input QR Code
3  set data from QR Code
4  set decoded from base64 decode data
5  return decoded
```

Pseudo*code* 4.1 Menu QR Code Scanner

4.4 Implementasi Menu GPS dan Google API

Implementasi menu GPS dan Google API yang dimaksud merupakan menu sebagai pembanding kinerja sebelum menggunaan dan sesudah menggunakan Google API.

```
1  input menu
2  if menu = "GPS" then
3      set layout as GPS
4  else if menu = "api" then
5      set layout as api
6  return layout
```

Pseudo*code* 4.2 Menu GPS dan Google API

Dengan menu GPS dan Google API kita dapat mengubah layout aplikasi sesuai dengan yang kita butuhkan sebagai perbandingan data hasil uji coba.

4.5 Implementasi GPSFragment

Untuk mengubah tampilan atau *layout* menjadi menu GPS, maka diperlukan implementasi terhadap GPSFragment. Implementasi ini akan menghasilkan tampilan untuk mencari data lokasi dengan memanfaatkan *framework* Android Location.

```
1  initialize constructor with parameter LayoutInflater, ViewGroup,
   and Bundle
2  set myView to GPS_layout in inflater
3  set textView to result in myView
4  set myList to listGPSSearch in myView
5  set data to ArrayList
6  set adapter to ArrayAdapter
```

```
7   set searchButton to buttonSearchLocation in myView
8   if GPS = null then
9       set GPS to GPSTracker
10      run GPS.getLocation
11  else
12      run GPS.stopUsingGPS
13      set GPS to null
```

Pseudo*code* 4.3 Kelas GPSFragment

Tampilan *layout* akan berubah sesuai dengan kelas yang dijalankan tersebut. *Layout* akan berubah dengan transaksi dari *fragment*.

4.6 Implementasi GPSTracker

Pada bagian ini berisi *pseudocode* yang menjelaskan tentang kerja kelas yang mengatur pengambilan data lokasi via GPS dengan *framework* Android Location. Selanjutnya, perhitungan dilakukan oleh ketiga metode untuk menentukan lokasi mahasiswa.

```
1   initialize constructor with parameter Context, TextView , ListView,
    ArrayList<String>
2   set context to Context in parameter
3   set myList to ListView in parameter
4   set data to ArrayList in parameter
5   set textView to TextView in parameter
6   set adapter to ArrayAdapter in myList
7   set dataPlaces from query syntax to local database
8   set locationManager to LocationManager in context
9   set locationManager.getBestProvider to criteria
10  set location to locationManager.getLastKnownLocation
11  set longitude to location.getLongitude
```

12	**set** latitude to location.getLatitude
13	**set** accuracy to location.getAccuracy
14	**set** altitude to location.getAltitude
15	**set** dataIteration with longitude, latitude, accuracy, altitude in array
16	**do** KNN method
17	**do** *Simple Average* method
18	**do** regression method
19	**sort** distance with ascending from KNN, *Simple Average*, regression
20	**push** distance to *database*
21	**set** textView.text to ID in dataPlaces

Pseudo*code* 4.4 Kelas GPSTracker

Kelas GPSTracker akan melakukan pencarian dengan memanfaatkan *framework* Android Location. GPS akan digabungkan dengan kriteria yang diset, sehingga akan memperhitungkan penggabungan jaringan, penggunaan baterai, dan criteria lain.

4.7 Implementasi GoogleAPI*Fragment*

Untuk mengubah tampilan atau *layout* menjadi menu googleAPI, maka diperlukan implementasi terhadap GoogleAPI. Implementasi ini akan menghasilkan tampilan untuk mencari data lokasi dengan memanfaatkan *framework* Google API.

1	**initialize** constructor with parameter LayoutInflater, ViewGroup, and Bundle
2	**set** myView to GPS_layout in inflater
3	**set** textView to result in myView
4	**set** myList to listGPSSearch in myView
5	**set** data to ArrayList
6	**set** adapter to ArrayAdapter
7	**set** searchAPIButton to buttonSearchAPILocation in myView

```
8   if googleAPI = null then
9       set googleAPI to GoogleAPITracker
10      run googelAPI.getLocation
11  else
12      run googleAPI.stopUsingAPI
13      set googleAPI to null
```

Pseudo*code* 4.5 Kelas GoogleAPIFragment

Tampilan *layout* akan berubah sesuai dengan kelas yang dijalankan tersebut. *Layout* akan berubah dengan transaksi dari *fragment* menjadi tampilan untuk mengambil lokasi dengan Google API.

4.8 Implementasi Google API *Tracker*

Pada bagian ini berisi *pseudocode* yang menjelaskan tentang kerja kelas yang mengatur pengambilan data lokasi via GPS dengan *framework* Google API. Selanjutnya, perhitungan dilakukan oleh ketiga metode untuk menentukan lokasi mahasiswa.

```
1   initialize constructor with parameter Context, TextView , ListView,
    ArrayList<String>
2   set context to Context in parameter
3   set myList to ListView in parameter
4   set data to ArrayList in parameter
5   set textView to TextView in parameter
6   set adapter to ArrayAdapter in myList
7   set result to GooglePlayServicesUtil.isGooglePlayServicesAvailable
    with context
8   if result != “Success” then
9       if result = “recoverable” then
11          set dataPlaces from query syntax to local database
12          set googleAPIClient to GoogleApiClient.Builder with context
```

13	**do** googleAPIClient.connect
	do requestUpdateLocation
10	**do** FusedLocationAPI with googleAPIClient
11	**set** longitude to location.getLongitude
12	**set** latitude to location.getLatitude
13	**set** accuracy to location.getAccuracy
14	**set** altitude to location.getAltitude
15	**set** dataIteration with longitude, latitude, accuracy, altitude in array
16	**do** KNN method
17	**do** *Simple Average* method
18	**do** regression method
19	**sort** distance with ascending from KNN, *Simple Average*, regression
20	**push** distance to *database*
	set textView.text to ID in dataPlaces
	else
	print "This device is not supported"

Pseudo*code* 4.6 Kelas GoogleAPITracker

Kelas GoogleAPITracker akan melakukan pencarian dengan memanfaatkan *framework* Google API. Sebelum menjalankan fungsi pencarian lokasi, terlebih dahul aplikasi akan mengecek kompatibilitas dari *smartphone*. Apakah *smartphone* tersebut telah memiliki Google Play Services, agar aplikasi dapat berjalan. Selanjutnya, apabila aplikasi telah memastikan keberadaan Google Play Service akan menjalankan Google API Client untuk digunakan sebagai *fused location* atau penggabungan sensor, jaringan, baterai, dan kriteria lainnya untuk melakukan pencarian lokasi. Data lokasi yang berhasil didapatkan tiap iterasinya (tergantung dari jumlah iterasi), akan diproses oleh ketiga metode. Ketiga metode tersebut akan mencari jarak terpendek sebagai hasil akhir untuk ditampilkan dan akan menvalidasi kehadiran mahasiswa.

4.9 Implementasi Metode KNN

Pada bagian ini berisi *pseudocode* yang menjelaskan tentang implementasi algoritma KNN yang mencari kedekatan mahasiswa dengan lokasi yang ada pada *database*.

```
1   input dataPlaces in array from database
2   input dataIteration in array from location
3   initialize result as Array
4   initialize temp as Array
5   foreach iteration in dataIteration then
6       set loc1 to Location with longitude, latitude, altitude in iteration
7       foreach place in dataPlaces then
8           set loc2 to Location with longitude, latitude, altitude in place
9           distance = loc1.getDistance from loc2
10          push distance, ID from place to temp
11      sort temp with distance ascending order
12      push first data in distance to result
13  push result to server
```

Pseudo*code* 4.7 Implementasi Metode KNN

Algoritma dimulai setelah data lokasi ruangan dan data lokasi mahasiswa telah didapatkan. Selanjutnya, variable result disiapkan untuk menampung hasil perhitungan jarak yang sudah diurutkan, sementara temp digunakan untuk menampung hasil perhitungan sebelum diurutkan. Masing-masing data lokasi mahasiswa akan dibandingkan dengan data lokasi ruangan satu per satu. Setiap hasil perhitungan akan disimpan sementara pada variabel array temp. Setelah masing-masing data lokasi sudah dihitung jaraknya dengan semua data lokasi ruangan, maka temp akan diurutkan sesuai dengan jarak terkecil. Hasil teratas pengurutan, akan dimasukkan ke variabel result. Data result ini

yang akan dilanjutkan menuju *server* untuk tahap kehadiran validasi mahasiswa.

4.10 Implementasi Metode Rata-Rata

Pada bagian ini berisi *pseudocode* yang menjelaskan tentang implementasi algoritma *Simple Average* atau rata-rata yang mencari kedekatan mahasiswa dengan lokasi yang ada pada *database*.

```
1  input dataPlaces in array from database
2  input dataIteration in array from location
3  initialize result in Array
4  initialize Simple AverageLongitude in Double
5  initialize Simple AverageLatitude in Double
6  initialize Simple AverageAltitude in Double
7  set length to Length in dataIteration
8  foreach iteration in dataIteration then
9      set longitude to longitude in iteration
10     set latitude to latitude in iteration
11     set altitude to altitude in iteration
12     Simple AverageLongitude = Simple AverageLongitude +
       longitude
13     Simple AverageLatitude = Simple AverageLatitude + latitude
14     Simple AverageAltitude = Simple AverageAltitude + altitude
15 Simple AverageLongitude = Simple AverageLongitude / length
16 Simple AverageLatitude = Simple AverageLatitude / length
17 Simple AverageAltitude = Simple AverageAltitude / length
18 set loc1 to Location with longitude, latitude, altitude
19 foreach place in dataPlaces then
20     set loc2 to Location with longitude, latitude, altitude in place
21     distance = loc1.getDistance from loc2
```

22	**push** distance, ID from place to temp
23	**sort** temp with distance ascending order
24	**push** first data in distance to result
25	**push** result to *server*

Pseudo*code* 4.8 Implementasi Metode Rata-Rata

Algoritma dimulai setelah data lokasi ruangan dan data lokasi mahasiswa telah didapatkan. Selanjutnya, variable result disiapkan untuk menampung hasil perhitungan jarak yang sudah diurutkan, sementara temp digunakan untuk menampung hasil perhitungan sebelum diurutkan. Masing-masing data lokasi ruangan akan dibandingkan dengan data rata-rata *longitude, latitude, dan altitude* pada data lokasi mahasiswa. Setiap hasil perhitungan akan disimpan sementara pada variabel array temp. Setelah masing-masing data lokasi sudah dihitung jaraknya dengan semua data lokasi ruangan, maka temp akan diurutkan sesuai dengan jarak terkecil. Hasil teratas pengurutan, akan dimasukkan ke variabel result. Data result ini yang akan dilanjutkan menuju *server* untuk tahap kehadiran validasi mahasiswa.

4.11 Implementasi Metode Regresi Linier

Pada bagian ini berisi *pseudocode* yang menjelaskan tentang implementasi algoritma Regresi Linier yang menghasilkan persamaan garis untuk mencari kedekatan mahasiswa dengan lokasi yang ada pada *database*.

1	**input** dataPlaces in array from *database*
2	**input** dataIteration in array from location
3	sumX = 0, sumY = 0, sumX2 = 0, sumY2 = 0, sumXY = 0, medianX, medianY, a, b, resultX, resultY
4	**initialize** result in Array
5	**initialize** sumX in Double
6	**initialize** sumY in Double

```
7   initialize Simple AverageAltitude in double
8   initialize sumX2 in Double
9   initialize sumY2 in Double
10  initialize sumXY in Double
11  initialize medianX in Double
12  initialize medianY in Double
13  initialize a in Double
14  initialize b in Double
15  initialize resultX in Double
16  initialize resultY in Double
17  initialize mid in Integer
17  set length to Length in dataIteration
18  foreach iteration in dataIteration then
19      set x to longitude in iteration
20      set y to latitude in iteration
21      set altitude to altitude in iteration
22      sumX = sumX + x
23      sumY = sumY + y
24      Simple AverageAltitude = Simple AverageAltitude + altitude
25      sumX2 = sumX2 + (x ^ 2)
26      sumY2 = sumY2 + (y ^ 2)
27      sumXY = sumXY + (x * y)
28  mid = length / 2
29  if length % 2 = 0 then
30      medianX= longitude in dataIteration[mid] + longitude in
        dataIteration[mid + 1]
31      medianY= latitude in dataIteration[mid] + latitude in
        dataIteration[mid + 1]
32  Else
33      medianX= longitude in dataIteration[mid]
34      medianY= latitude in dataIteration[mid]
```

35	b = ((length * sumXY) - (sumX * sumY)) / ((length * sumX2) - (sumX * sumX))
36	a = (sumY / length) - (b * (sumX / length))
37	resultX = (medianY - a) / b
38	resultY = a + (b * medianX)
39	**set** loc1 to Location with resultX, resultY, *Simple Average*Altitude
40	**foreach** place in dataPlaces then
41	**set** loc2 to Location with longitude, latitude, altitude in place
42	distance = loc1.getDistance from loc2
43	**push** distance, ID from place to temp
44	**sort** temp with distance ascending order
45	**push** first data in distance to result
46	**push** result to *server*

Pseudo*code* 4.9 Implementasi Metode Regresi Linier

Algoritma dimulai setelah data lokasi ruangan dan data lokasi mahasiswa telah didapatkan. Selanjutnya, variable result disiapkan untuk menampung hasil perhitungan jarak yang sudah diurutkan.

Langkah-langkah algoritma ini:

1. Menyimpan hasil penjumlahan semua longitude yang ada pada data lokasi mahasiswa ke variabel sumX
2. Menyimpan hasil penjumlahan semua *latitude* yang ada pada data lokasi mahasiswa ke variabel sumY.
3. Menyimpan hasil penjumlahan semua *altitude* yang ada pada data lokasi mahasiswa ke variabel *Simple Average*Altitude.
4. Menghitung nilai sumX2 dengan menjumlahan keseluruhan *longitude* pada data lokasi mahasiswa dengan setiap datanya merupakan perpangkatan 2.
5. Menghitung nilai sumY2 dengan menjumlahan keseluruhan *latitude* pada data lokasi mahasiswa dengan setiap datanya merupakan perpangkatan 2.

6. Menghitung nilai sumXY dengan menjumlahan keseluruhan perkalian *longitude* dan *latitude* pada setiap iterasi data lokasi mahasiswa.
7. Mencari data tengah dari *longitude* dan disimpan ke medianX
8. Mencari data tengah dari *latitude* dan disimpan ke medianY
9. Mencari nilai a, b sesuai dengan rumus yang telah dijelaskan pada bab sebelumnya.
10. Menghitung nilai x baru sesuai persamaan dan disimpan pada variabel resultX
11. Menghitung nilai y baru sesuai persamaan dan disimpan pada variabel resultY
12. Melakukan perbandingan setiap data lokasi ruangan.

Setelah masing-masing data lokasi sudah dihitung jaraknya dengan semua data lokasi ruangan, maka selanjutnya hasil akan diurutkan sesuai dengan jarak terkecil. Hasil teratas pengurutan, akan dilanjutkan menuju *server* untuk tahap kehadiran validasi mahasiswa.

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Bab ini menjelaskan uji coba dari aplikasi yang telah berhasil dikerjakan, serta pembahasan hasil uji coba. Pembahasan hasil uji coba meliputi evaluasi dari penggunaan ketiga metode dan bagaimana tingkat akurasi terhadap banyaknya percobaan yang dilakukan oleh tiap pengguna.

5.1 Kondisi Uji Coba

Kondisi uji coba menjelaskan tentang kondisi yang harus dipenuhi untuk menjalankan uji coba aplikasi. Kondisi uji coba merupakan pemilihan pengguna dan *smartphone* yang dijelaskan sebagai berikut:

Tabel 5.1 Spesifikasi Smartphone untuk Uji Coba

No	NRP	Smartphone	Seri Android	RAM	Seri GPS
1	5113100051	Sony C5	OS 6 Marshmallow	2 GB	A-GPS
2	5113100075	Samsung Note 3	OS 6 Marshmallow	3 GB	A-GPS
3	5113100128	Sony Xperia Z5	OS 7 Nougat	3 GB	A-GPS
4	5113100062	Xiaomi Redmi Note 2	OS 5 Lolipop	2 GB	A-GPS
5	5113100072	Oppo A37	OS 5 Lolipop	2 GB	A-GPS
6	5113100143	Samsung J2	OS 5 Lolipop	1 GB	A-GPS
7	5113100173	Sony Xperia Z1 Compact	OS 5 Lolipop	2 GB	A-GPS

5.2 Data Lokasi Ruangan Uji Coba

Pada tugas akhir ini, penulis memetakan lokasi ruangan yang berada pada gedung Teknik Informatika ITS dengan mencari titik tengah dari ruangan. Lokasi ruangan ini yang akan digunakan sebagai data pembanding atau klasifikasi.

5.3 Skenario dan Evaluasi Pengujian

Uji coba ini dilakukan untuk menguji apakah fungsionalitas program telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian dan kinerja aplikasi.

Skenario pengujian terdiri dari dua skenario pengujian, yaitu:

1. Skenario melakukan pencarian terhadap lokasi ruangan IF-101 yang dilakukan di laboratorium Rekayasa Perangkat Lunak Teknik Informatika ITS.
2. Skenario melakukan pencarian terhadap lokasi ruangan IF-103 yang dilakukan di laboratorium Komputasi Cerdas dan Visualisasi.

5.3.1 Skenario Uji Coba

Skenario uji coba 1 mewajibkan mahasiswa untuk melakukan pemindaian terhadap *QR Code* yang berisi kalimat "IF-101 Informatika Sikemas" dengan Base64. Percobaan akan dilakukan sebanyak empat kali, yaitu:

a. Percobaan dilakukan oleh mahasiswa dengan NRP 5113100051.
b. Uji coba selanjutnya dilakukan oleh mahasiswa dengan NRP 5113100075.
c. Kemudian, uji coba selanjutnya dilakukan oleh mahasiswa dnegan NRP 5113100128.
d. Uji coba terakhir dilakukan oleh mahasiswa dengan NRP 5115100062.

Sementara, skenario uji coba 2 mewajibkan mahasiswa untuk melakukan pemindaian terhadap *QR Code* yang berisi kalimat "IF-102 Informatika Sikemas" dengan MD5 Hash. Percobaan akan dilakukan sebanyak tiga kali, yaitu:

a. Percobaan dilakukan oleh mahasiswa dengan NRP 5113100072.
b. Uji coba selanjutnya dilakukan oleh mahasiswa dengan NRP 5113100143.
c. Kemudian, uji coba selanjutnya dilakukan oleh mahasiswa dnegan NRP 5113100173.

5.3.1.1 Skenario Uji Coba 1 NRP 5113100051

Skenario uji coba 1 ini dilakukan oleh mahasiswa dengan NRP 5113100051 di laboratorium Rekayasa Perangkat Lunak dengan mengacu kepada ruangan IF-101.

5.3.1.1.1 Percobaan Ke-1

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-1 untuk skenario uji coba 1 NRP 5113100051.

5.3.1.1.1.1 Iterasi Data Lokasi

Tabel 5.2 Hasil Iterasi Data Lokasi Percobaan Ke-1 NRP 5113100051

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972087	-7.2793511	0	15.65799999
2	112.7967665	-7.2792739	-47	9
3	112.7968336	-7.2792754	-18	11
4	112.7968611	-7.2792815	-12	13
5	112.7970356	-7.2793249	1	13
6	112.7970217	-7.2793337	9	12

7	112.7970259	-7.2793495	5	12
8	112.7970476	-7.2793586	13	12
9	112.7970627	-7.2793605	19	12
10	112.7970831	-7.2793682	17	12

Tabel 5.2 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.1.1.2 Hasil KNN

Tabel 5.3 Hasil Metode KNN Percobaan Ke-1 NRP 5113100051

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	3.162420988	10

Tabel 5.3 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-1.

5.3.1.1.1.3 Hasil Rata-Rata (Simple Average)

Tabel 5.4 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100051

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7969946	-7.27932773	-1.3

Tabel 5.5 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	14.61829281

2	3	IF-102	22.34718895
3	4	IF-103	28.97241592
4	5	IF-104	36.70206070
5	6	IF-105a	43.32756042
6	7	IF-105b	49.95311737
7	8	IF-106	66.83567047
8	10	IF-108	76.72232819
9	12	Lab Pemrograman 2	68.03355408
10	13	Algoritma Pemrograman	57.96274567
11	14	Manajemen Informasi	53.78499603
12	15	Dasar dan Terapan Komunikasi	52.15001678

Tabel 5.4 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-1 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.5 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.1.4 Hasil Regresi Linier

Tabel 5.6 Hasil SUM Metode Regresi Linier Percobaan Ke-1 NRP 5113100051

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.969946	-72.7932773	127231.62	529.886122	-8210.86291

Tabel 5.6 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.7 Hasil Perhitungan Metode Regresi Linier Percobaan 1 NRP 5113100051

MEDIANX	MEDIANY	A	B
112.7970287	-7.2793293	20.90149273	-0.249836625

Tabel 5.7 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.8 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-1 NRP 5113100051

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7970009	-7.279336224	-1.3

Tabel 5.8 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-1 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.9 Hasil Metode Regresi Linier Percobaan Ke-1 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	13.97646618
2	3	IF-102	21.68676186
3	4	IF-103	28.30413246
4	5	IF-104	36.02826691
5	6	IF-105a	42.65063477
6	7	IF-105b	49.27389908
7	8	IF-106	66.00460815
8	10	IF-108	75.64731598
9	12	Lab Pemrograman 2	66.86960602
10	13	Algoritma Pemrograman	56.80741501
11	14	Manajemen Informasi	52.67531967

12	15	Dasar dan Terapan Komunikasi	51.07147980

Tabel 5.9 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.2 Percobaan Ke-2

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-2 untuk skenario uji coba 1 NRP 5113100051.

5.3.1.1.2.1 Iterasi Data Lokasi

Tabel 5.10 Hasil Iterasi Data Lokasi Percobaan Ke-2 NRP 5113100051

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972067	-7.2793495	0	12.88099957
2	112.7969684	-7.2793028	33	14
3	112.7969687	-7.2792997	33	12
4	112.7969716	-7.2792967	31	9
5	112.796972	-7.2792937	30	7
6	112.7969757	-7.2792903	30	6
7	112.7969794	-7.2792879	29	6
8	112.7969863	-7.2792876	28	6
9	112.7969913	-7.2792881	28	6
10	112.7969965	-7.2792890	27	6

Tabel 5.10 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-2. Data ini yang nantinya akan

diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.1.2.2 Hasil KNN

Tabel 5.11 Hasil Metode KNN Percobaan Ke-2 NRP 5113100051

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	2.913553715	10

Tabel 5.11 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-2.

5.3.1.1.2.3 Hasil Rata-Rata (Simple Average)

Tabel 5.12 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-2 NRP 5113100051

*SIMPLE AVERAGE*LONGITUDE	*SIMPLE AVERAGE*LATITUDE	*SIMPLE AVERAGE*ALTITUDE
112.7970017	-7.27929853	26.9

Tabel 5.13 Hasil Metode Rata-Rata Percobaan Ke-2 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	14.14725494
2	3	IF-102	21.76877594
3	4	IF-103	28.34829330
4	5	IF-104	36.04578400
5	6	IF-105a	42.65298462
6	7	IF-105b	49.26515198
7	8	IF-106	66.67372131
8	10	IF-108	77.77056122

9	12	Lab Pemrograman 2	70.00850677
10	13	Algoritma Pemrograman	60.48639679
11	14	Manajemen Informasi	56.62980652
12	15	Dasar dan Terapan Komunikasi	55.11392212

Tabel 5.12 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-2 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.13 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.2.4 Hasil Regresi Linier

Tabel 5.14 Hasil SUM Metode Regresi Linier Percobaan Ke-2 NRP 5113100051

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.970017	-72.7929853	127231.6358	529.8818709	-8210.830484

Tabel 5.14 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-2.

Tabel 5.15 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-2 NRP 5113100051

MEDIANX	MEDIANY	A	B
112.7969739	-7.279292	19.39398853	-0.236471596

Tabel 5.15 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-2.

Tabel 5.16 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-2 NRP 5113100051

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.796974	-7.279291954	26.9

Tabel 5.16 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-2 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.17 Hasil Metode Regresi Linier Percobaan Ke-2 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	17.28133583
2	3	IF-102	24.89021873
3	4	IF-103	31.45914459
4	5	IF-104	39.14708328
5	6	IF-105a	45.74802017
6	7	IF-105b	52.35521317
7	8	IF-106	69.80786896
8	10	IF-108	80.73014832
9	12	Lab Pemrograman 2	72.51560211
10	13	Algoritma Pemrograman	62.52483749
11	14	Manajemen Informasi	58.26570129
12	15	Dasar dan Terapan Komunikasi	56.55609894

Tabel 5.17 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-2. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.3 Percobaan Ke-3

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-3 untuk skenario uji coba 1 NRP 5113100051.

5.3.1.1.3.1 Iterasi Data Lokasi

Tabel 5.18 Hasil Iterasi Data Lokasi Percobaan Ke-3 NRP 5113100051

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972029	-7.2793465	0	15.6960001
2	112.7970225	-7.2792977	33	9
3	112.7970219	-7.2792979	30	6
4	112.797021	-7.2792985	29	8
5	112.7970212	-7.2793006	27	7
6	112.7970232	-7.2793029	26	5
7	112.7970245	-7.2793060	25	4
8	112.7970265	-7.2793103	24	5
9	112.7970283	-7.2793161	23	5
10	112.7970298	-7.2793209	24	4

Tabel 5.18 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-3. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.1.3.2 Hasil KNN

Tabel 5.19 Hasil Metode KNN Percobaan Ke-3 NRP 5113100051

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	2.465377808	10

Tabel 5.19 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-3.

5.3.1.1.3.3 Hasil Rata-Rata (Simple Average)

Tabel 5.20 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-3 NRP 5113100051

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7970422	-7.27930974	24.1

Tabel 5.21 Hasil Metode Rata-Rata Percobaan Ke-3 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	9.517343521
2	3	IF-102	17.17959023
3	4	IF-103	23.78214455
4	5	IF-104	31.49740982
5	6	IF-105a	38.11523819
6	7	IF-105b	44.73539352
7	8	IF-106	62.03926468
8	10	IF-108	73.35717010
9	12	Lab Pemrograman 2	66.28767395
10	13	Algoritma Pemrograman	57.51421356
11	14	Manajemen Informasi	54.29912949

12	15	Dasar dan Terapan Komunikasi	53.09089279

Tabel 5.20 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-3 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.21 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.3.4 Hasil Regresi Linier

Tabel 5.22 Hasil SUM Metode Regresi Linier Percobaan Ke-3 NRP 5113100051

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.970422	-72.7930974	127231.7272	529.8835029	-8210.846078

Tabel 5.22 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-3.

Tabel 5.23 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-3 NRP 5113100051

MEDIANX	MEDIANY	A	B
112.7970222	-7.27930175	19.26720514	-0.235347615

Tabel 5.23 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-3.

Tabel 5.24 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-3 NRP 5113100051

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7970082	-7.279305038	24.1

Tabel 5.24 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-3 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.25 Hasil Metode Regresi Linier Percobaan Ke-3 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	13.30003929
2	3	IF-102	20.96208763
3	4	IF-103	27.55976868
4	5	IF-104	35.27033997
5	6	IF-105a	41.88509369
6	7	IF-105b	48.50283432
7	8	IF-106	65.81565857
8	10	IF-108	76.77021027
9	12	Lab Pemrograman 2	68.99282837
10	13	Algoritma Pemrograman	59.52227020
11	14	Manajemen Informasi	55.73412704
12	15	Dasar dan Terapan Komunikasi	54.25669098

Tabel 5.25 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-3. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.4 Percobaan Ke-4

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-4 untuk skenario uji coba 1 NRP 5113100051.

5.3.1.1.4.1 Iterasi Data Lokasi

Tabel 5.26 Hasil Iterasi Data Lokasi Percobaan Ke-4 NRP 5113100051

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7970232	-7.2793029	26	5
2	112.7970245	-7.2793060	25	4
3	112.7970265	-7.2793103	24	5
4	112.7970283	-7.2793161	23	5
5	112.7970298	-7.2793209	24	4
6	112.7970304	-7.2793262	23	5
7	112.7970309	-7.2793298	22	4
8	112.7970317	-7.2793323	22	4
9	112.7970321	-7.2793339	21	4
10	112.7970328	-7.2793348	21	4

Tabel 5.26 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-4. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.1.4.2 Hasil KNN

Tabel 5.27 Hasil Metode KNN Percobaan Ke-4 NRP 5113100051

NO	ID	NAME	DISTANCE	COUNTER
1	2	IF-101	11.72012329	10

Tabel 5.27 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-4.

5.3.1.1.4.3 Hasil Rata-Rata (Simple Average)

Tabel 5.28 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-4 NRP 5113100051

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.797029	-7.27932132	23.1

Tabel 5.29 Hasil Metode Rata-Rata Percobaan Ke-4 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	10.82740784
2	3	IF-102	18.55419540
3	4	IF-103	25.17871475
4	5	IF-104	32.90791702
5	6	IF-105a	39.53318787
6	7	IF-105b	46.15858078
7	8	IF-106	63.21741867
8	10	IF-108	73.86506653
9	12	Lab Pemrograman 2	66.15383911
10	13	Algoritma Pemrograman	56.91863632
11	14	Manajemen Informasi	53.39172745

12	15	Dasar dan Terapan Komunikasi	52.0532341

Tabel 5.28 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-4 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.29 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.4.4 Hasil Regresi Linier

Tabel 5.30 Hasil SUM Metode Regresi Linier Percobaan Ke-4 NRP 5113100051

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.97029	-72.7932132	127231.6976	529.8851888	-8210.858182

Tabel 5.30 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-4.

Tabel 5.31 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-4 NRP 5113100051

MEDIANX	MEDIANY	A	B
112.7970301	-7.27932355	415.7095375	-3.75

Tabel 5.31 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-4.

Tabel 5.32 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-4 NRP 5113100051

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7970296	-7.27932537	23.1

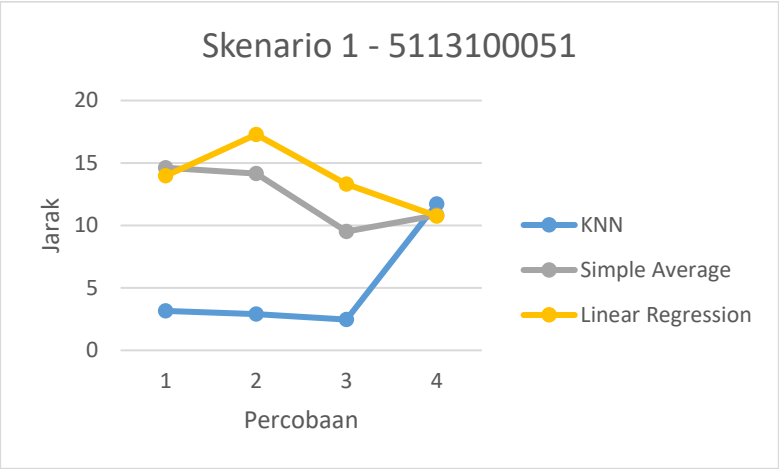
Tabel 5.32 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-4 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.33 Hasil Metode Regresi Linier Percobaan Ke-4 NRP 5113100051

NO	ID	NAME	DISTANCE
1	2	IF-101	10.75416660
2	3	IF-102	18.48410988
3	4	IF-103	25.10979080
4	5	IF-104	32.83975601
5	6	IF-105a	39.46544647
6	7	IF-105b	46.09113693
7	8	IF-106	63.07305527
8	10	IF-108	73.56871033
9	12	Lab Pemrograman 2	65.76187897
10	13	Algoritma Pemrograman	56.48376465
11	14	Manajemen Informasi	52.94156265
12	15	Dasar dan Terapan Komunikasi	51.60071564

Tabel 5.33 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100051 pada skenario uji coba 1, percobaan ke-4. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.1.5 Evaluasi



Grafik 5.1 Kesimpulan Skenario Uji Coba 1 Mahasiswa NRP 5113100051

Grafik 5.1 menunjukkan hasil ketiga metode selama percobaan hingga ke-n dilakukan. Percobaan terakhir menunjukkan kesamaan dari ketiga metode.

5.3.1.2 Skenario Uji Coba 1 NRP 5113100075

Skenario uji coba 1 ini dilakukan oleh mahasiswa dengan NRP 5113100075 di laboratorium Rekayasa Perangkat Lunak dengan mengacu kepada ruangan IF-101.

5.3.1.2.1 Percobaan Ke-1

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-1 untuk skenario uji coba 1 NRP 5113100075.

5.3.1.2.1.1 Iterasi Data Lokasi

Tabel 5.34 Hasil Iterasi Data Lokasi Percobaan Ke-1 NRP 5113100075

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7970358	-7.2793325	25	19
2	112.7970362	-7.2793322	25	14
3	112.7970363	-7.2793318	25	11
4	112.7970361	-7.2793312	25	10
5	112.7970359	-7.2793311	25	10
6	112.7970357	-7.2793319	25	9
7	112.7970355	-7.2793327	24	8
8	112.7970351	-7.2793328	24	7
9	112.797035	-7.2793327	24	7
10	112.7970348	-7.2793329	24	6

Tabel 5.34 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100075 pada skenario uji coba 1, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.2.1.2 Hasil KNN

Tabel 5.35 Hasil Metode KNN Percobaan Ke-1 NRP 5113100075

NO	ID	NAME	DISTANCE	COUNTER
1	2	IF-101	10.10515118	10

Tabel 5.35 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100075 pada skenario uji coba 1, percobaan ke-1.

5.3.1.2.1.3 Hasil Rata-Rata (Simple Average)

Tabel 5.36 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100075

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7970356	-7.27933218	24.6

Tabel 5.37 Hasil Metode Rata-Rata Percobaan Ke-1 NRP 5113100075

NO	ID	NAME	DISTANCE
1	2	IF-101	10.11992168
2	3	IF-102	17.83638000
3	4	IF-103	24.45728302
4	5	IF-104	32.18416214
5	6	IF-105a	38.80818176
6	7	IF-105b	45.43268967
7	8	IF-106	62.28949356
8	10	IF-108	72.60382080
9	12	Lab Pemrograman 2	64.75833130
10	13	Algoritma Pemrograman	55.52312088
11	14	Manajemen Informasi	52.04847336
12	15	Dasar dan Terapan Komunikasi	50.74707413

Tabel 5.36 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100075 pada skenario uji coba 1, percobaan ke-1 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.37 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.2.1.4 Hasil Regresi Linier

Tabel 5.38 Hasil SUM Metode Regresi Linier Percobaan Ke-1 NRP 5113100075

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.970356	-72.7933218	127231.7125	529.8867699	-8210.870913

Tabel 5.38 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100075 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.39 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-1 NRP 5113100075

MEDIANX	MEDIANY	A	B
112.7970358	-7.2793315	13.870112	-0.1875

Tabel 5.39 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100075 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.40 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-1 NRP 5113100075

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.797032	-7.27933221	24.6

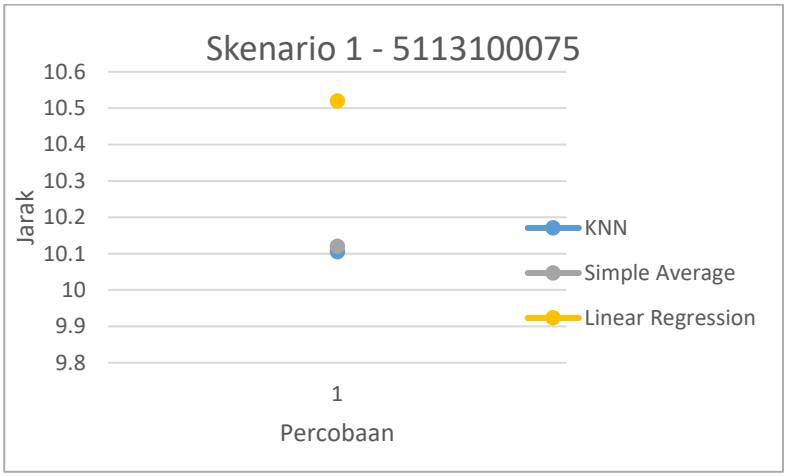
Tabel 5.40 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100075 pada skenario uji coba 1, percobaan ke-1 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.41 Hasil Metode Regresi Linier Percobaan Ke-1 NRP 5113100075

NO	ID	NAME	DISTANCE
1	2	IF-101	10.51947117
2	3	IF-102	18.23662376
3	4	IF-103	24.85766792
4	5	IF-104	32.58460999
5	6	IF-105a	39.20865250
6	7	IF-105b	45.83317184
7	8	IF-106	62.68399429
8	10	IF-108	72.94126892
9	12	Lab Pemrograman 2	65.00532532
10	13	Algoritma Pemrograman	55.68567276
11	14	Manajemen Informasi	52.14302826
12	15	Dasar dan Terapan Komunikasi	50.81014633

Tabel 5.41 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100075 pada skenario uji coba 1, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.2.2 Evaluasi



Grafik 5.2 Kesimpulan Skenario Uji Coba 1 Mahasiswa NRP 5113100075

Grafik 5.2 menunjukkan hasil ketiga metode selama percobaan hingga ke-n dilakukan. Percobaan terakhir menunjukkan kesamaan dari ketiga metode.

5.3.1.3 Skenario Uji Coba 1 NRP 5113100128

Skenario uji coba 1 ini dilakukan oleh mahasiswa dengan NRP 5113100128 di laboratorium Rekayasa Perangkat Lunak dengan mengacu kepada ruangan IF-101.

5.3.1.3.1 Percobaan Ke-1

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-1 untuk skenario uji coba 1 NRP 5113100128.

5.3.1.3.1.1 Iterasi Data Lokasi

Tabel 5.42 Hasil Iterasi Data Lokasi Percobaan Ke-1 NRP 5113100128

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972204	-7.2793504	0	15.86499977
2	112.7970533	-7.2793389	43	18
3	112.7970544	-7.2793392	38	12
4	112.7970557	-7.2793400	34	8
5	112.7970571	-7.2793407	31	7
6	112.7970579	-7.2793409	31	7
7	112.7970581	-7.2793408	31	9
8	112.7970586	-7.2793412	31	10
9	112.7970595	-7.2793417	31	11
10	112.7970596	-7.2793419	31	11

Tabel 5.42 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.3.1.2 Hasil KNN

Tabel 5.43 Hasil Metode KNN Percobaan Ke-1 NRP 5113100128

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	3.816787004	10

Tabel 5.43 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-1.

5.3.1.3.1.3 Hasil Rata-Rata (Simple Average)

Tabel 5.44 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100128

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7970735	-7.27934157	30.1

Tabel 5.45 Hasil Metode Rata-Rata Percobaan Ke-1 NRP 5113100128

NO	ID	NAME	DISTANCE
1	2	IF-101	6.189806461
2	3	IF-102	13.76482773
3	4	IF-103	20.35066795
4	5	IF-104	28.05787468
5	6	IF-105a	34.67212296
6	7	IF-105b	41.29003906
7	8	IF-106	58.00003815
8	10	IF-108	68.52961731
9	12	Lab Pemrograman 2	61.40599060
10	13	Algoritma Pemrograman	52.97016144
11	14	Manajemen Informasi	50.18042755
12	15	Dasar dan Terapan Komunikasi	49.20558167

Tabel 5.44 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-1 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.45 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.3.1.4 Hasil Regresi Linier

Tabel 5.46 Hasil SUM Metode Regresi Linier Percobaan Ke-1 NRP 5113100128

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.970735	-72.7934157	127231.7978	529.8881369	-8210.884258

Tabel 5.46 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.47 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-1 NRP 5113100128

MEDIANX	MEDIANY	A	B
112.7970575	-7.2793408	-0.434461022	-0.06068314

Tabel 5.47 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.48 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-1 NRP 5113100128

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7970608	-7.279340601	30.1

Tabel 5.48 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-1 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.49 Hasil Metode Regresi Linier Percobaan Ke-1 NRP 5113100128

NO	ID	NAME	DISTANCE
1	2	IF-101	7.514307022
2	3	IF-102	15.14213371
3	4	IF-103	21.73778725
4	5	IF-104	29.44976425
5	6	IF-105a	36.06616592
6	7	IF-105b	42.68544769
7	8	IF-106	59.40055847
8	10	IF-108	69.76258850
9	12	Lab Pemrograman 2	62.32471848
10	13	Algoritma Pemrograman	53.57792282
11	14	Manajemen Informasi	50.53738403
12	15	Dasar dan Terapan Komunikasi	49.44670868

Tabel 5.49 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.3.2 Percobaan Ke-2

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-2 untuk skenario uji coba 1 NRP 5113100128.

5.3.1.3.2.1 Iterasi Data Lokasi

Tabel 5.50 Hasil Iterasi Data Lokasi Percobaan Ke-2 NRP 5113100128

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7970557	-7.2793400	34	8

2	112.7970571	-7.2793407	31	7
3	112.7970579	-7.2793409	31	7
4	112.7970581	-7.2793408	31	9
5	112.7970586	-7.2793412	31	10
6	112.7970595	-7.2793417	31	11
7	112.7970596	-7.2793419	31	11
8	112.7970601	-7.2793422	30	12
9	112.7970604	-7.2793425	29	11
10	112.7970607	-7.2793426	29	11

Tabel 5.50 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-2. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.3.2.2 Hasil KNN

Tabel 5.51 Hasil Metode KNN Percobaan Ke-2 NRP 5113100128

NO	ID	NAME	DISTANCE	COUNTER
1	2	IF-101	8.04638958	10

Tabel 5.51 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-2.

5.3.1.3.2.3 Hasil Rata-Rata (Simple Average)

Tabel 5.52 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-2 NRP 5113100128

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7970588	-7.27934145	30.8

Tabel 5.53 Hasil Metode Rata-Rata Percobaan Ke-2 NRP 5113100128

NO	ID	NAME	DISTANCE
1	2	IF-101	7.751037598
2	3	IF-102	15.37251949
3	4	IF-103	21.96565056
4	5	IF-104	29.67597580
5	6	IF-105a	36.29148102
6	7	IF-105b	42.91012573
7	8	IF-106	59.60418701
8	10	IF-108	69.89790344
9	12	Lab Pemrograman 2	62.38298035
10	13	Algoritma Pemrograman	53.57395554
11	14	Manajemen Informasi	50.48845673
12	15	Dasar dan Terapan Komunikasi	49.37827301

Tabel 5.52 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-2 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.53 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.3.2.4 Hasil Regresi Linier

Tabel 5.54 Hasil SUM Metode Regresi Linier Percobaan Ke-2 NRP 5113100128

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.970588	-72.7934145	127231.7647	529.8881195	-8210.883053

Tabel 5.54 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-2.

Tabel 5.55 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-2 NRP 5113100128

MEDIANX	MEDIANY	A	B
112.7970591	-7.27934145	20.91992324	-0.25

Tabel 5.55 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-2.

Tabel 5.56 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-2 NRP 5113100128

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7970588	-7.27934152	30.8

Tabel 5.56 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-2 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

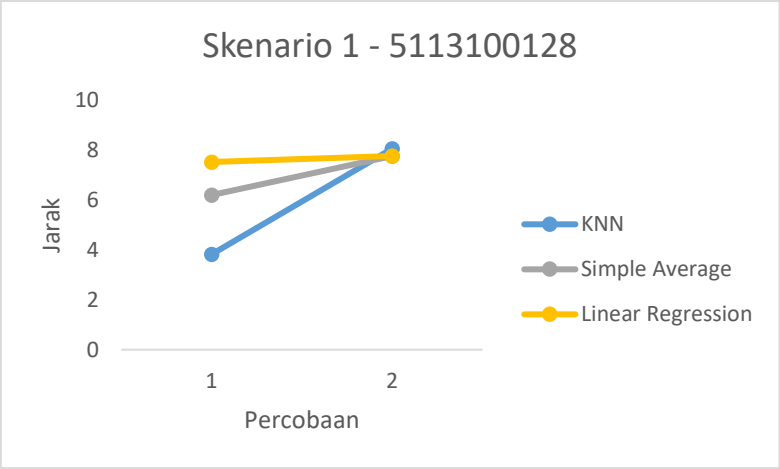
Tabel 5.57 Hasil Metode Regresi Linier Percobaan Ke-2 NRP 5113100128

NO	ID	NAME	DISTANCE
1	2	IF-101	7.752858162
2	3	IF-102	15.37343693
3	4	IF-103	21.96629333
4	5	IF-104	29.67645073
5	6	IF-105a	36.29186630
6	7	IF-105b	42.91045761
7	8	IF-106	59.60298538
8	10	IF-108	69.89373016
9	12	Lab Pemrograman 2	62.37681198
10	13	Algoritma Pemrograman	53.56678772

11	14	Manajemen Informasi	50.48086548
12	15	Dasar dan Terapan Komunikasi	49.37058258

Tabel 5.57 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100128 pada skenario uji coba 1, percobaan ke-2. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.3.3 Evaluasi



Grafik 5.3 Kesimpulan Skenario Uji Coba 1 Mahasiswa NRP 5113100128

Grafik 5.3 menunjukkan hasil ketiga metode selama percobaan hingga ke-n dilakukan. Percobaan terakhir menunjukkan kesamaan dari ketiga metode.

5.3.1.4 Skenario Uji Coba 1 NRP 5113100062

Skenario uji coba 1 ini dilakukan oleh mahasiswa dengan NRP 5113100062 di laboratorium Rekayasa Perangkat Lunak dengan mengacu kepada ruangan IF-101.

5.3.1.4.1 Percobaan Ke-1

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-1 untuk skenario uji coba 1 NRP 5113100062.

5.3.1.4.1.1 Iterasi Data Lokasi

Tabel 5.58 Hasil Iterasi Data Lokasi Percobaan Ke-1 NRP 5113100062

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7970914	-7.2795925	54	20
2	112.7970915	-7.2795750	53	14
3	112.797091	-7.2795573	52	10
4	112.7970913	-7.2795478	51	9
5	112.7970907	-7.2795402	51	8
6	112.797089	-7.2795316	50	7
7	112.7970876	-7.2795247	51	7
8	112.7970865	-7.2795167	51	6
9	112.7970859	-7.2795111	51	6
10	112.7970854	-7.2795068	51	6

Tabel 5.58 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.4.1.2 Hasil KNN

Tabel 5.59 Hasil Metode KNN Percobaan Ke-1 NRP 5113100062

NO	ID	NAME	DISTANCE	COUNTER
1	15	Dasar dan Terapan Komunikasi	21.38783455	8
2	2	IF-101	21.07567978	2

Tabel 5.59 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1.

5.3.1.4.1.3 Hasil Rata-Rata (Simple Average)

Tabel 5.60 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100062

*SIMPLE AVERAGE*LONGITUDE	*SIMPLE AVERAGE*LATITUDE	*SIMPLE AVERAGE*ALTITUDE
112.797089	-7.27954037	51.5

Tabel 5.61 Hasil Metode Rata-Rata Percobaan Ke-1 NRP 5113100062

NO	ID	NAME	DISTANCE
1	2	IF-101	24.18446922
2	3	IF-102	26.63576508
3	4	IF-103	30.18872452
4	5	IF-104	35.46649551
5	6	IF-105a	40.62018967
6	7	IF-105b	46.15113068
7	8	IF-106	56.98870850
8	10	IF-108	57.70834732

9	12	Lab Pemrograman 2	44.13043594
10	13	Algoritma Pemrograman	32.36508179
11	14	Manajemen Informasi	28.28275490
12	15	Dasar dan Terapan Komunikasi	27.15307045

Tabel 5.60 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.61 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.4.1.4 Hasil Regresi Linier

Tabel 5.62 Hasil SUM Metode Regresi Linier Percobaan Ke-1 NRP 5113100062

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.97089	-72.7954037	127231.8329	529.91708	-8211.109632

Tabel 5.62 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.63 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-1 NRP 5113100062

MEDIANX	MEDIANY	A	B
112.7970898	-7.2795359	2685.75096	-23.875

Tabel 5.62 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1.

Tabel 5.64 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-1 NRP 5113100062

RESULTX	RESULTY	*SIMPLE AVERAGE* ALTITUDE
112.7970888	-7.279559947	51.5

Tabel 5.64 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.65 Hasil Metode Regresi Linier Percobaan Ke-1 NRP 5113100062

NO	ID	NAME	DISTANCE
1	2	IF-101	26.32277679
2	3	IF-102	28.59691620
3	4	IF-103	31.93669891
4	5	IF-104	36.97006226
5	6	IF-105a	41.94266129
6	7	IF-105b	47.32217407
7	8	IF-106	57.53213882
8	10	IF-108	57.17877960
9	12	Lab Pemrograman 2	42.82031250
10	13	Algoritma Pemrograman	30.55061150
11	14	Manajemen Informasi	26.18797112
12	15	Dasar dan Terapan Komunikasi	24.99844742

Tabel 5.65 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.4.2 Percobaan Ke-2

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-2 untuk skenario uji coba 1 NRP 5113100062.

5.3.1.4.2.1 Iterasi Data Lokasi

Tabel 5.66 Hasil Iterasi Data Lokasi Percobaan Ke-2 NRP 5113100062

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972029	-7.2793465	0	15.54500008
2	112.7970781	-7.2794523	49	14
3	112.7970777	-7.2794507	49	7
4	112.7970459	-7.2794494	46	5
5	112.7970201	-7.2794657	4	6
6	112.7970062	-7.2794674	2	5
7	112.7969809	-7.2794758	-11	6
8	112.796956	-7.2794856	-24	5
9	112.7969536	-7.2794904	-28	4
10	112.7969518	-7.2794900	-19	5

Tabel 5.66 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.4.2.2 Hasil KNN

Tabel 5.67 Hasil Metode KNN Percobaan Ke-2 NRP 5113100062

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	2.465377808	10

Tabel 5.67 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-2.

5.3.1.4.2.3 Hasil Rata-Rata (Simple Average)

Tabel 5.68 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-2 NRP 5113100062

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7970273	-7.27945738	6.8

Tabel 5.69 Hasil Metode Rata-Rata Percobaan Ke-2 NRP 5113100062

NO	ID	NAME	DISTANCE
1	2	IF-101	18.31664848
2	3	IF-102	23.77871323
3	4	IF-103	29.28522301
4	5	IF-104	36.18687057
5	6	IF-105a	42.33106613
6	7	IF-105b	48.60193634
7	8	IF-106	62.45932770
8	10	IF-108	67.12247467
9	12	Lab Pemrograman 2	55.27184677
10	13	Algoritma Pemrograman	43.76248550
11	14	Manajemen Informasi	39.05858612

12	15	Dasar dan Terapan Komunikasi	37.36484909

Tabel 5.68 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-2 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.69 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.4.2.4 Hasil Regresi Linier

Tabel 5.70 Hasil SUM Metode Regresi Linier Percobaan Ke-2 NRP 5113100062

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.970273	-72.7945738	127231.6937	529.9049975	-8211.01153

Tabel 5.70 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-2.

Tabel 5.71 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-2 NRP 5113100062

MEDIANX	MEDIANY	A	B
112.7970132	-7.27946655	-65.07126202	0.512352196

Tabel 5.71 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-2.

Tabel 5.72 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke- NRP 5113100062

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7970094	-7.27946464	6.8

Tabel 5.72 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-2 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.73 Hasil Metode Regresi Linier Percobaan Ke-2 NRP 5113100062

NO	ID	NAME	DISTANCE
1	2	IF-101	20.17599487
2	3	IF-102	25.83706284
3	4	IF-103	31.39970207
4	5	IF-104	38.31921387
5	6	IF-105a	44.46326447
6	7	IF-105b	50.72870255
7	8	IF-106	64.48220825
8	10	IF-108	68.67732239
9	12	Lab Pemrograman 2	56.24017715
10	13	Algoritma Pemrograman	44.20798111
11	14	Manajemen Informasi	39.04287338
12	15	Dasar dan Terapan Komunikasi	37.12938690

Tabel 5.73 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-2. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.4.3 Percobaan Ke-3

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-3 untuk skenario uji coba 1 NRP 5113100062.

5.3.1.4.3.1 Iterasi Data Lokasi

Tabel 5.74 Hasil Iterasi Data Lokasi Percobaan Ke-3 NRP 5113100062

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7969432	-7.2794736	-11	6
2	112.7969439	-7.2794614	-8	5
3	112.7969455	-7.2794561	-9	4
4	112.7969517	-7.2794517	-8	4
5	112.7969594	-7.2794457	-7	4
6	112.796963	-7.2794468	-9	4
7	112.7969655	-7.2794484	-9	5
8	112.7969699	-7.2794467	-9	5
9	112.7969762	-7.2794430	-6	6
10	112.7969809	-7.2794407	-6	7

Tabel 5.74 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-3. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.4.3.2 Hasil KNN

Tabel 5.75 Hasil KNN Percobaan Ke-3 NRP 5113100062

NO	ID	NAME	DISTANCE	COUNTER
1	2	IF-101	26.11574364	10

Tabel 5.75 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-3.

5.3.1.4.3.3 Hasil Rata-Rata (Simple Average)

Tabel 5.76 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-3 NRP 5113100062

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7969599	-7.27945141	-8.2

Tabel 5.77 Hasil Metode Rata-Rata Percobaan Ke-3 NRP 5113100062

NO	ID	NAME	DISTANCE
1	2	IF-101	23.14852333
2	3	IF-102	29.67908287
3	4	IF-103	35.66052628
4	5	IF-104	42.87895584
5	6	IF-105a	49.19006729
6	7	IF-105b	55.57445526
7	8	IF-106	69.86053467
8	10	IF-108	74.31206512
9	12	Lab Pemrograman 2	61.41775513
10	13	Algoritma Pemrograman	48.70782471
11	14	Manajemen Informasi	42.77675247
12	15	Dasar dan Terapan Komunikasi	40.44399643
13	2	IF-101	23.14852333

Tabel 5.76 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-3 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.77 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.4.3.4 Hasil Regresi Linier

Tabel 5.78 Hasil SUM Metode Regresi Linier Percobaan Ke-3 NRP 5113100062

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.969599	-72.7945141	127231.5417	529.9041283	-8210.999889

Tabel 5.78 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-3.

Tabel 5.79 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-3 NRP 5113100062

MEDIANX	MEDIANY	A	B
112.7969612	-7.27944625	-79.08680179	0.636607143

Tabel 5.79 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-3.

Tabel 5.80 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-3 NRP 5113100062

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.796968	-7.279450595	-8.2

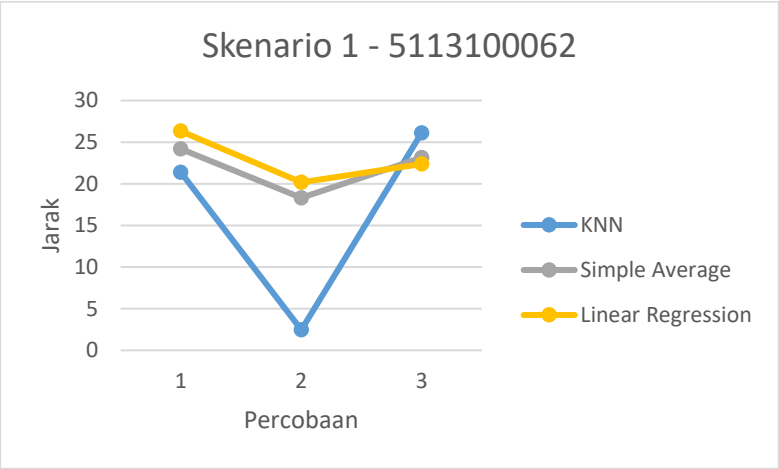
Tabel 5.80 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-3 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.81 Hasil Metode Regresi Linier Percobaan Ke-3 NRP 5113100062

NO	ID	NAME	DISTANCE
1	2	IF-101	22.38559723
2	3	IF-102	28.84910774
3	4	IF-103	34.80280685
4	5	IF-104	42.00391769
5	6	IF-105a	48.30658340
6	7	IF-105b	54.68566895
7	8	IF-106	68.96249390
8	10	IF-108	73.50331879
9	12	Lab Pemrograman 2	60.76777649
10	13	Algoritma Pemrograman	48.21045685
11	14	Manajemen Informasi	42.42653656
12	15	Dasar dan Terapan Komunikasi	40.16848755

Tabel 5.81 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100062 pada skenario uji coba 1, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.4.4 Evaluasi



Grafik 5.4 Kesimpulan Skenario Uji Coba 1 Mahasiswa NRP 5113100062

Grafik 5.4 menunjukkan hasil ketiga metode selama percobaan hingga ke-n dilakukan. Percobaan terakhir menunjukkan kesamaan dari ketiga metode.

5.3.1.5 Skenario Uji Coba 2 NRP 5113100072

Skenario uji coba 2 ini dilakukan oleh mahasiswa dengan NRP 5113100072 di laboratorium Komputasi Cerdas dan Visualisasi dengan mengacu kepada ruangan IF-103.

5.3.1.5.1 Percobaan Ke-1

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah

terpenuhi. Percobaan ini merupakan percobaan ke-1 untuk skenario uji coba 2 NRP 5113100072.

5.3.1.5.1.1 Iterasi Data Lokasi

Tabel 5.82 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-1 NRP 5113100072

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7971739	-7.2793114	49	17
2	112.7973058	-7.2793896	43	10
3	112.7973139	-7.2793992	43	8
4	112.7972676	-7.2793717	45	7
5	112.7972425	-7.2793607	45	7
6	112.7972388	-7.2793667	46	5
7	112.7972467	-7.2793676	43	5
8	112.7972468	-7.2793634	42	4
9	112.7972339	-7.2793639	43	4
10	112.7972155	-7.2793597	45	4

Tabel 5.82 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.5.1.2 Hasil KNN

Tabel 5.83 Hasil Metode KNN Percobaan Ke-1 NRP 5113100072

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	2.961287498	10

Tabel 5.83 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-1.

5.3.1.5.1.3 Hasil Rata-Rata (Simple Average)

Tabel 5.84 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100072

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7972485	-7.27936539	44.4

Tabel 5.85 Hasil Metode Rata-Rata Linier Percobaan Ke-1 NRP 5113100072

NO	ID	NAME	DISTANCE
1	2	IF-101	14.14523315
2	3	IF-102	7.235002995
3	4	IF-103	4.563464642
4	5	IF-104	9.747859001
5	6	IF-105a	15.92900372
6	7	IF-105b	22.36616135
7	8	IF-106	38.49633408
8	10	IF-108	51.61905670
9	12	Lab Pemrograman 2	49.95239258
10	13	Algoritma Pemrograman	46.96406555
11	14	Manajemen Informasi	48.12018585
12	15	Dasar dan Terapan Komunikasi	48.86231613

Tabel 5.84 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-1 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.85 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.5.1.4 Hasil Regresi Linier

Tabel 5.86 Hasil SUM Metode Regresi Linier Percobaan Ke-1 NRP 5113100072

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.972485	-72.7936539	127232.1928	529.8916048	-8210.923871

Tabel 5.86 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-1.

Tabel 5.87 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-1 NRP 5113100072

MEDIANX	MEDIANY	A	B
112.7972406	-7.2793637	53.05174108	-0.534863281

Tabel 5.87 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-1.

Tabel 5.88 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-1 NRP 5113100072

RESULTX	RESULTY	*SIMPLE AVERAGE* ALTITUDE
112.7972454	-7.27936117	44.4

Tabel 5.88 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-1 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.89 Hasil Metode Regresi Linier Percobaan Ke-1 NRP 5113100072

NO	ID	NAME	DISTANCE
1	2	IF-101	13.67083836

2	3	IF-102	6.674108505
3	4	IF-103	4.200871468
4	5	IF-104	9.860888481
5	6	IF-105a	16.14228058
6	7	IF-105b	22.62098312
7	8	IF-106	38.92207718
8	10	IF-108	52.19211578
9	12	Lab Pemrograman 2	50.50928116
10	13	Algoritma Pemrograman	47.42668915
11	14	Manajemen Informasi	48.49690247
12	15	Dasar dan Terapan Komunikasi	49.19863129

Tabel 5.89 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.5.2 Percobaan Ke-2

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-2 untuk skenario uji coba 2 NRP 5113100072.

5.3.1.5.2.1 Iterasi Data Lokasi

Tabel 5.90 Hasil Iterasi Data Lokasi Percobaan Ke-2 NRP 5113100072

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7971972	-7.2793619	51	18
2	112.797197	-7.2793618	51	18
3	112.7971969	-7.2793620	50	14
4	112.7971968	-7.2793623	51	12

5	112.7971969	-7.2793624	51	11
6	112.7971999	-7.2793658	51	7
7	112.7972067	-7.2793685	50	7
8	112.7972081	-7.2793687	49	6
9	112.7972088	-7.2793691	49	5
10	112.7972099	-7.2793698	48	5

Tabel 5.90 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-2. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.5.2.2 Hasil KNN

Tabel 5.91 Hasil Metode KNN Percobaan Ke-2 NRP 5113100072

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	4.080908298	10

Tabel 5.91 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-2.

5.3.1.5.2.3 Hasil Rata-Rata (Simple Average)

Tabel 5.92 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-2 NRP 5113100072

*SIMPLE AVERAGE*LONGITUDE	*SIMPLE AVERAGE*LATITUDE	*SIMPLE AVERAGE*ALTITUDE
112.7972018	-7.27936523	50.1

Tabel 5.93 Hasil Metode Rata-Rata Percobaan Ke-2 NRP 5113100072

NO	ID	NAME	DISTANCE
1	2	IF-101	9.383989334
2	3	IF-102	4.480845451
3	4	IF-103	7.544835567
4	5	IF-104	14.52174759
5	6	IF-105a	20.92749786
6	7	IF-105b	27.43790817
7	8	IF-106	43.59096146
8	10	IF-108	55.53140640
9	12	Lab Pemrograman 2	51.92403030
10	13	Algoritma Pemrograman	47.18528748
11	14	Manajemen Informasi	47.22719193
12	15	Dasar dan Terapan Komunikasi	47.51765823

Tabel 5.92 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-2 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.93 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.5.2.4 Hasil Regresi Linier

Tabel 5.94 Hasil SUM Metode Regresi Linier Percobaan Ke-2 NRP 5113100072

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.972018	-72.7936523	127232.0874	529.8915815	-8210.92029

Tabel 5.94 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-2.

Tabel 5.95 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-2 NRP 5113100072

MEDIANX	MEDIANY	A	B
112.7971984	-7.2793641	63.80637133	-0.630208333

Tabel 5.95 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-2.

Tabel 5.96 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-2 NRP 5113100072

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7972	-7.279363075	50.1

Tabel 5.96 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-2 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.97 Hasil Metode Regresi Linier Percobaan Ke-2 NRP 5113100072

NO	ID	NAME	DISTANCE
1	2	IF-101	9.097382545
2	3	IF-102	4.224006176
3	4	IF-103	7.570510387
4	5	IF-104	14.64002609
5	6	IF-105a	21.07209587
6	7	IF-105b	27.59593773
7	8	IF-106	43.82330322
8	10	IF-108	55.83544159
9	12	Lab Pemrograman 2	52.22369766
10	13	Algoritma Pemrograman	47.44153214
11	14	Manajemen Informasi	47.43981171

12	15	Dasar dan Terapan Komunikasi	47.70890045

Tabel 5.97 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-2. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.5.3 Percobaan Ke-3

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-3 untuk skenario uji coba 2 NRP 5113100072.

5.3.1.5.3.1 Iterasi Data Lokasi

Tabel 5.98 Hasil Iterasi Data Lokasi Percobaan Ke-3 NRP 5113100072

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7971905	-7.2793483	45	16
2	112.7971894	-7.2793476	45	13
3	112.7971871	-7.2793459	46	10
4	112.7971805	-7.2793423	46	9
5	112.7971817	-7.2793435	46	9
6	112.7971765	-7.2793418	47	7
7	112.7971757	-7.2793413	47	6
8	112.7971741	-7.2793390	46	5
9	112.7971749	-7.2793363	45	5
10	112.7971744	-7.2793365	45	5

Tabel 5.98 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-3. Data ini yang nantinya akan

diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.5.3.2 Hasil KNN

Tabel 5.99 Hasil Metode KNN Percobaan Ke-3 NRP 5113100072

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-103	2.474900055	10

Tabel 5.99 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-3.

5.3.1.5.3.3 Hasil Rata-Rata (Simple Average)

Tabel 5.100 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-3 NRP 5113100072

SIMPLE AVERAGELONGITUDE	SIMPLE AVERAGELATITUDE	SIMPLE AVERAGEALTITUDE
112.7971805	-7.27934225	45.8

Tabel 5.101 Hasil Metode Rata-Rata Percobaan Ke-3 NRP 5113100072

NO	ID	NAME	DISTANCE
1	2	IF-101	6.206180573
2	3	IF-102	2.639573336
3	4	IF-103	8.662638664
4	5	IF-104	16.29201889
5	6	IF-105a	22.88528633
6	7	IF-105b	29.49309158
7	8	IF-106	46.35572815
8	10	IF-108	58.96509933
9	12	Lab Pemrograman 2	55.23218536

10	13	Algoritma Pemrograman	49.98146057
11	14	Manajemen Informasi	49.52916718
12	15	Dasar dan Terapan Komunikasi	49.57935333

Tabel 5.100 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-3 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.101 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.5.3.4 Hasil Regresi Linier

Tabel 5.102 Hasil SUM Metode Regresi Linier Percobaan Ke-3 NRP 5113100072

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.971805	-72.7934225	127232.0392	529.8882359	-8210.892815

Tabel 5.102 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-3.

Tabel 5.103 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-3 NRP 5113100072

MEDIANX	MEDIANY	A	B
112.7971791	-7.27934265	62.33766758	-0.6171875

Tabel 5.103 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-3.

Tabel 5.104 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-3 NRP 5113100072

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7971811	-7.279341398	45.8

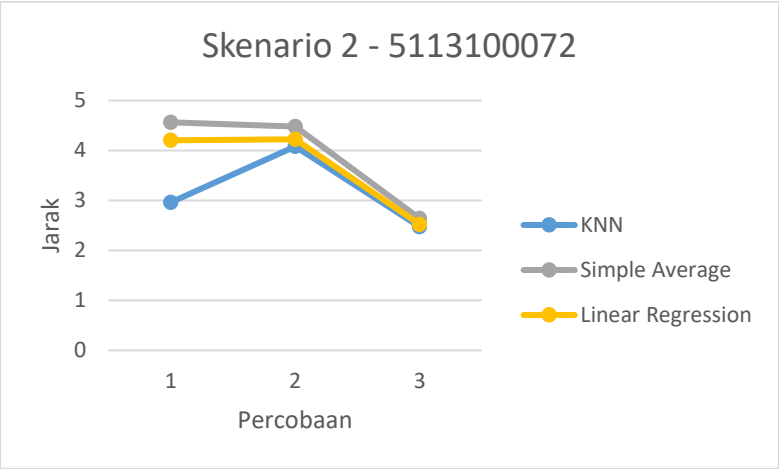
Tabel 5.104 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-3 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.105 Hasil Metode Regresi Linier Percobaan Ke-3 NRP 5113100072

NO	ID	NAME	DISTANCE
1	2	IF-101	6.246327400
2	3	IF-102	2.522069216
3	4	IF-103	8.572422028
4	5	IF-104	16.21013641
5	6	IF-105a	22.80628395
6	7	IF-105b	29.41571426
7	8	IF-106	46.30428696
8	10	IF-108	58.97008514
9	12	Lab Pemrograman 2	55.28551102
10	13	Algoritma Pemrograman	50.06512070
11	14	Manajemen Informasi	49.62819290
12	15	Dasar dan Terapan Komunikasi	49.68364334

Tabel 5.105 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100072 pada skenario uji coba 2, percobaan ke-3. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.5.4 Evaluasi



Grafik 5.5 Kesimpulan Skenario Uji Coba 2 Mahasiswa NRP 5113100072

Grafik 5.5 menunjukkan hasil ketiga metode selama percobaan hingga ke-n dilakukan. Percobaan terakhir menunjukkan kesamaan dari ketiga metode.

5.3.1.6 Skenario Uji Coba 2 NRP 5113100143

Skenario uji coba 2 ini dilakukan oleh mahasiswa dengan NRP 5113100143 di laboratorium Komputasi Cerdas dan Visualisasi dengan mengacu kepada ruangan IF-103.

5.3.1.6.1 Percobaan Ke-1

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-1 untuk skenario uji coba 2 NRP 5113100143.

5.3.1.6.1.1 Iterasi Data Lokasi

Tabel 5.106 Hasil Iterasi Data Lokasi Percobaan Ke-1 NRP 5113100143

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972976	-7.2793909	0	17.99500084
2	112.7973144	-7.2793936	0	14.23200035
3	112.7973223	-7.2793949	0	12.07900047
4	112.7973727	-7.2793830	0	11.01599979
5	112.7974084	-7.2793774	0	10.1960001
6	112.7974353	-7.2793707	0	10
7	112.7973826	-7.2792936	16	16
8	112.7973526	-7.2792976	20	16
9	112.7973492	-7.2793151	12	10
10	112.7971948	-7.2793408	-14	18

Tabel 5.106 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.6.1.2 Hasil KNN

Tabel 5.107 Hasil Metode KNN Percobaan Ke-1 NRP 5113100143

NO	ID	NAME	DISTANCE	COUNTER
1	5	IF-104	7.978444099	4
2	6	IF-105a	6.605862141	4
3	7	IF-105b	5.216589451	1
4	3	IF-102	1.764163017	1

Tabel 5.107 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1.

5.3.1.6.1.3 Hasil Rata-Rata (Simple Average)

Tabel 5.108 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100143

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.797343	-7.27935576	3.4

Tabel 5.109 Hasil Metode Rata-Rata Percobaan Ke-1 NRP 5113100143

NO	ID	NAME	DISTANCE
1	2	IF-101	24.09276009
2	3	IF-102	16.47641754
3	4	IF-103	10.08667755
4	5	IF-104	3.832779169
5	6	IF-105a	5.932234287
6	7	IF-105b	11.97882366
7	8	IF-106	28.54401207
8	10	IF-108	45.36363983
9	12	Lab Pemrograman 2	48.54405975
10	13	Algoritma Pemrograman	49.30140305
11	14	Manajemen Informasi	52.48379135
12	15	Dasar dan Terapan Komunikasi	53.99460602

Tabel 5.108 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.109 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.6.1.4 Hasil Regresi Linier

Tabel 5.110 Hasil SUM Metode Regresi Linier Percobaan Ke-1 NRP 5113100143

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.97343	-72.7935576	127232.4059	529.8902028	-8210.919884

Tabel 5.110 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1.

Tabel 5.111 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-1 NRP 5113100143

MEDIANX	MEDIANY	A	B
112.7974219	-7.27937405	-9.876663	0.023026316

Tabel 5.111 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1.

Tabel 5.112 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-1 NRP 5113100143

RESULTX	RESULTY	*SIMPLE AVERAGE* ALTITUDE
112.7965487	-7.279353944	3.4

Tabel 5.112 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.113 Hasil Metode Regresi Linier Percobaan Ke-1 NRP 5113100143

NO	ID	NAME	DISTANCE
1	2	IF-101	63.94285583

2	3	IF-102	71.66418457
3	4	IF-103	78.28382111
4	5	IF-104	86.00791168
5	6	IF-105a	92.62934875
6	7	IF-105b	99.25134277
7	8	IF-106	115.4794693
8	10	IF-108	120.7891388
9	12	Lab Pemrograman 2	105.7074203
10	13	Algoritma Pemrograman	90.48375702
11	14	Manajemen Informasi	81.91197205
12	15	Dasar dan Terapan Komunikasi	78.19684601

Tabel 5.113 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.6.2 Percobaan Ke-2

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-2 untuk skenario uji coba 2 NRP 5113100143.

5.3.1.6.2.1 Iterasi Data Lokasi

Tabel 5.114 Hasil Iterasi Data Lokasi Percobaan Ke-2 NRP 5113100143

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972495	-7.2793833	0	17.48699951
2	112.7973726	-7.2792900	55	19
3	112.7973781	-7.2792990	56	14
4	112.797368	-7.2793124	53	10

5	112.7973655	-7.2793117	52	8
6	112.7973651	-7.2793115	51	7
7	112.7973629	-7.2793113	52	7
8	112.7973617	-7.2793152	52	7
9	112.7973578	-7.2793188	51	6
10	112.7973584	-7.2793210	42	5

Tabel 5.114 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-2. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.6.2.2 Hasil KNN

Tabel 5.115 Hasil Metode KNN Percobaan Ke-2 NRP 5113100143

NO	ID	NAME	DISTANCE	COUNTER
1	4	IF-103	6.500495911	1
2	6	IF-105a	4.184630394	9

Tabel 5.115 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-2.

5.3.1.6.2.3 Hasil Rata-Rata (Simple Average)

Tabel 5.116 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-2 NRP 5113100143

*SIMPLE AVERAGE*LONGITUDE	*SIMPLE AVERAGE*LATITUDE	*SIMPLE AVERAGE*ALTITUDE
112.797354	-7.27931742	46.4

Tabel 5.117 Hasil Metode Rata-Rata Percobaan Ke-2 NRP 5113100143

NO	ID	NAME	DISTANCE
1	2	IF-101	25.07680321
2	3	IF-102	17.35307312
3	4	IF-103	3.092914104
4	5	IF-104	3.743612051
5	6	IF-105a	10.73988533
6	7	IF-105b	10.30838299
7	8	IF-106	28.85345078
8	10	IF-108	48.13155365
9	12	Lab Pemrograman 2	52.62039566
10	13	Algoritma Pemrograman	53.70607758
11	14	Manajemen Informasi	56.85250092
12	15	Dasar dan Terapan Komunikasi	58.30744934

Tabel 5.116 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-2 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.117 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.6.2.4 Hasil Regresi Linier

Tabel 5.118 Hasil SUM Metode Regresi Linier Percobaan Ke-2 NRP 5113100143

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.97354	-72.7931742	127232.4306	529.884621	-8210.877436

Tabel 5.118 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-2.

Tabel 5.119 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-2 NRP 5113100143

MEDIANX	MEDIANY	A	B
112.7973653	-7.2793116	-80.25889036	0.646997207

Tabel 5.119 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-2.

Tabel 5.120 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-2 NRP 5113100143

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.797363	-7.279310083	46.4

Tabel 5.120 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-2 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

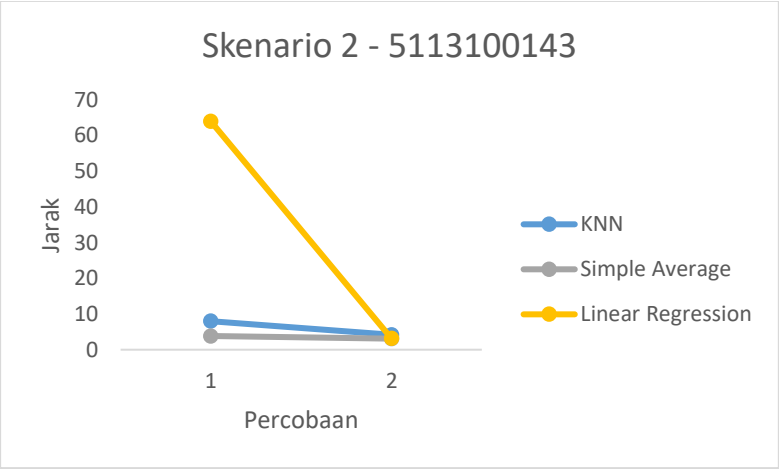
Tabel 5.121 Hasil Metode Regresi Linier Percobaan Ke-2 NRP 5113100143

NO	ID	NAME	DISTANCE
1	2	IF-101	26.10830688
2	3	IF-102	18.40026093
3	4	IF-103	3.125955343
4	5	IF-104	4.299569607
5	6	IF-105a	11.81619263
6	7	IF-105b	9.426373482
7	8	IF-106	28.30638313
8	10	IF-108	48.28561020
9	12	Lab Pemrograman 2	53.34270477
10	13	Algoritma Pemrograman	54.72992325
11	14	Manajemen Informasi	57.99527740

12	15	Dasar dan Terapan Komunikasi	59.48816681

Tabel 5.121 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100143 pada skenario uji coba 2, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.6.3 Evaluasi



Grafik 5.6 Kesimpulan Skenario Uji Coba 2 Mahasiswa NRP 5113100143

Grafik 5.6 menunjukkan hasil ketiga metode selama percobaan hingga ke-n dilakukan. Percobaan terakhir menunjukkan kesamaan dari ketiga metode.

5.3.1.7 Skenario Uji Coba 2 NRP 5113100173

Skenario uji coba 2 ini dilakukan oleh mahasiswa dengan NRP 5113100173 di laboratorium Komputasi Cerdas dan Visualisasi dengan mengacu kepada ruangan IF-103.

5.3.1.7.1 Percobaan Ke-1

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-1 untuk skenario uji coba 2 NRP 5113100173.

5.3.1.7.1.1 Iterasi Data Lokasi

Tabel 5.122 Hasil Iterasi Data Lokasi Percobaan Ke-1 NRP 5113100173

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7971660	-7.2793384	46	12
2	112.7971654	-7.2793371	45	8
3	112.7971659	-7.2793377	45	7
4	112.7971667	-7.2793380	44	5
5	112.7971677	-7.2793399	44	5
6	112.7971678	-7.2793402	44	5
7	112.7971698	-7.2793413	44	5
8	112.7971702	-7.2793412	44	4
9	112.7971694	-7.2793415	44	5
10	112.7971693	-7.2793417	44	5

Tabel 5.122 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-1. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.7.1.2 Hasil KNN

Tabel 5.123 Hasil Metode KNN Percobaan Ke-1 NRP 5113100173

NO	ID	NAME	DISTANCE	COUNTER
1	3	IF-102	3.730273247	10

Tabel 5.123 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-1.

5.3.1.7.1.3 Hasil Rata-Rata (Simple Average)

Tabel 5.124 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-1 NRP 5113100173

*SIMPLE AVERAGE*LONGITUDE	*SIMPLE AVERAGE*LATITUDE	*SIMPLE AVERAGE*ALTITUDE
112.7971678	-7.2793397	44.4

Tabel 5.125 Hasil Metode Rata-Rata Percobaan Ke-1 NRP 5113100173

NO	ID	NAME	DISTANCE
1	2	IF-101	4.791877270
2	3	IF-102	3.609169006
3	4	IF-103	9.981271744
4	5	IF-104	17.65298080
5	6	IF-105a	24.25819206
6	7	IF-105b	30.87218285
7	8	IF-106	47.78190994
8	10	IF-108	60.22600174
9	12	Lab Pemrograman 2	56.11059570
10	13	Algoritma Pemrograman	50.46884537

11	14	Manajemen Informasi	49.73247528
12	15	Dasar dan Terapan Komunikasi	49.65885162

Tabel 5.124 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-1 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.125 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.7.1.4 Hasil Regresi Linier

Tabel 5.126 Hasil SUM Metode Regresi Linier Percobaan Ke-1 NRP 5113100173

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.971678	-72.793397	127232.0107	529.8878647	-8210.889018

Tabel 5.126 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-1.

Tabel 5.127 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-1 NRP 5113100173

MEDIANX	MEDIANY	A	B
112.7971678	-7.27934005	63.21889019	-0.625

Tabel 5.127 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-1.

Tabel 5.128 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-1 NRP 5113100173

RESULTX	RESULTY	*SIMPLE AVERAGE*ALTITUDE
112.7971684	-7.279339656	44.4

Tabel 5.128 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-1 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.129 Hasil Metode Regresi Linier Percobaan Ke-1 NRP 5113100173

NO	ID	NAME	DISTANCE
1	2	IF-101	4.848475456
2	3	IF-102	3.551856518
3	4	IF-103	9.919470787
4	5	IF-104	17.59095764
5	6	IF-105a	24.19616699
6	7	IF-105b	30.81017303
7	8	IF-106	47.72224808
8	10	IF-108	60.18095779
9	12	Lab Pemrograman 2	56.08666992
10	13	Algoritma Pemrograman	50.46361923
11	14	Manajemen Informasi	49.73993301
12	15	Dasar dan Terapan Komunikasi	49.67163849

Tabel 5.129 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-1. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.7.2 Percobaan Ke-2

Setiap skenario uji coba akan menghasilkan percobaan. Banyaknya percobaan tergantung dari hasil yang didapatkan. Percobaan ke-n akan selesai apabila kondisi dari skenario telah terpenuhi. Percobaan ini merupakan percobaan ke-2 untuk skenario uji coba 2 NRP 5113100173.

5.3.1.7.2.1 Iterasi Data Lokasi

Tabel 5.130 Hasil Iterasi Data Lokasi Percobaan Ke-2 NRP 5113100173

NO	LONGITUDE	LATITUDE	ALTITUDE	ACCURACY
1	112.7972535	-7.2793803	0	20
2	112.7972365	-7.2793508	27	17
3	112.7972316	-7.2793417	20	15
4	112.7972317	-7.2793416	20	15
5	112.7972331	-7.2793402	19	15
6	112.7972290	-7.2793434	21	14
7	112.7972268	-7.2793540	25	13
8	112.7972255	-7.2793516	26	12
9	112.7972244	-7.2793551	28	12
10	112.7972248	-7.2793537	29	11

Tabel 5.130 merupakan hasil yang di dapat setelah GPS melakukan pencarian lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-2. Data ini yang nantinya akan diklasifikasikan dengan ketiga metode yang telah disebutkan pada bab sebelumnya.

5.3.1.7.2.2 Hasil KNN

Tabel 5.131 Hasil Metode KNN Percobaan Ke-2 NRP 5113100173

NO	ID	NAME	DISTANCE	COUNTER
1	4	IF-103	6.127943993	10

Tabel 5.131 merupakan hasil klasifikasi metode KNN dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-2.

5.3.1.7.2.3 Hasil Rata-Rata (Simple Average)

Tabel 5.132 Hasil Perhitungan Metode Rata-Rata Percobaan Ke-2 NRP 5113100173

***SIMPLE AVERAGE*LONGITUDE**	***SIMPLE AVERAGE*LATITUDE**	***SIMPLE AVERAGE*ALTITUDE**
112.7972317	-7.27935124	21.5

Tabel 5.133 Hasil Metode Rata-Rata Percobaan Ke-2 NRP 5113100173

NO	ID	NAME	DISTANCE
1	2	IF-101	11.91938400
2	3	IF-102	4.805821419
3	4	IF-103	4.029009342
4	5	IF-104	10.91764736
5	6	IF-105a	17.39438248
6	7	IF-105b	23.95273590
7	8	IF-106	40.61596680
8	10	IF-108	54.04785156
9	12	Lab Pemrograman 2	52.06447983
10	13	Algoritma Pemrograman	48.53674698
11	14	Manajemen Informasi	49.26641846
12	15	Dasar dan Terapan Komunikasi	49.81708908

Tabel 5.132 merupakan hasil klasifikasi metode SA dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-2 yang menunjukan rata-rata dari *longitude*, *latitude*, dan *altitude*. Sementara tabel 5.133 merupakan hasil perhitungan dari metode SA sebelum diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.7.2.4 Hasil Regresi Linier

Tabel 5.134 Hasil SUM Metode Regresi Linier Percobaan Ke-2 NRP 5113100173

SUMX	SUMY	SUMX2	SUMY2	SUMXY
1127.972317	-72.7935124	127232.1548	529.8895448	-8210.906684

Tabel 5.134 merupakan hasil penjumlahan dari variable yang dibutuhkan untuk perhitungan dengan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-2.

Tabel 5.135 Hasil Perhitungan Metode Regresi Linier Percobaan Ke-2 NRP 5113100173

MEDIANX	MEDIANY	A	B
112.7972311	-7.2793418	86.01335914	-0.827083333

Tabel 5.135 merupakan hasil perhitungan metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-2.

Tabel 5.136 Hasil Akhir Koordinat Metode Regresi Linier Percobaan Ke-2 NRP 5113100173

RESULTX	RESULTY	*SIMPLE AVERAGE* ALTITUDE
112.7972203	-7.279350711	21.5

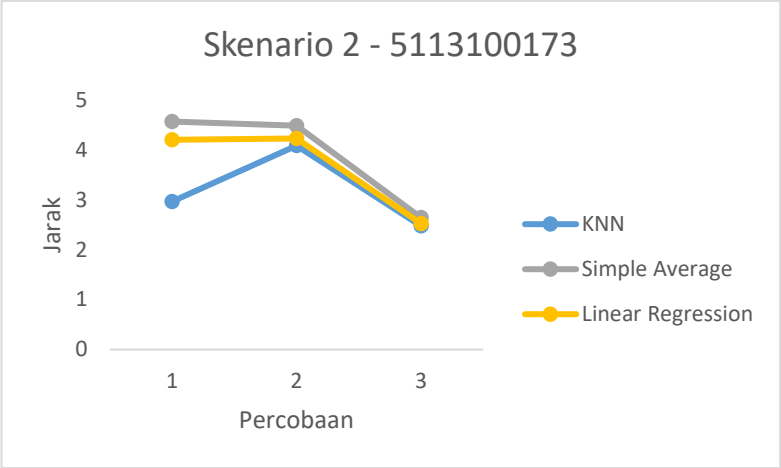
Tabel 5.136 merupakan hasil akhir metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-2 yang menentukan koordinat *longitude*, *latitude*, dan *altitude*.

Tabel 5.137 Hasil Metode Regresi Linier Percobaan Ke-2 NRP 5113100173

NO	ID	NAME	DISTANCE
1	2	IF-101	10.68559933
2	3	IF-102	4.952831268
3	4	IF-103	3.832976818
4	5	IF-104	12.12345314
5	6	IF-105a	18.62926292
6	7	IF-105b	25.19762611
7	8	IF-106	41.86288452
8	10	IF-108	55.02128983
9	12	Lab Pemrograman 2	52.58181381
10	13	Algoritma Pemrograman	48.64112473
11	14	Manajemen Informasi	49.10623550
12	15	Dasar dan Terapan Komunikasi	49.54872894

Tabel 5.137 merupakan hasil metode Regresi Linier dari data lokasi mahasiswa NRP 5113100173 pada skenario uji coba 2, percobaan ke-2. Hasil ini nantinya akan diurutkan berdasarkan jarak terpendek atau terkecil.

5.3.1.7.3 Evaluasi



Grafik 5.7 Kesimpulan Skenario Uji Coba 2 Mahasiswa NRP 5113100173

Grafik 5.7 menunjukkan hasil ketiga metode selama percobaan hingga ke-n dilakukan. Percobaan terakhir menunjukkan kesamaan dari ketiga metode.

5.3.2 Evaluasi Pengujian

Dari semua uji coba yang telah dilakukan, maka dapat disimpulkan:

1. Ketiga metode memberikan kode ruangan yang sama.
2. Jarak yang dihasilkan setiap metodenya memiliki selisih yang tidak jauh berbeda.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan pada bab sebelumnya, yaitu Bab Uji Coba dan Evaluasi yang merupakan jawaban dari latar belakang dan rumusan masalah pada bab sebelumnya. Setelah menjelaskan tentang kesimpulan yang diambil, bab ini juga memberikan saran untuk pengembangan aplikasi.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. Pengambilan data lokasi mahasiswa dilakukan pada beberapa laboratorium Teknik Informatika ITS dengan rata-rata percobaan sebanyak 2 kali.
2. Ketinggian atau *altitude* yang tidak pasti, sehingga tidak dapat digunakan untuk menentukan posisi mahasiswa secara pasti, bisa 0, bilangan negatif, atau bilangan puluhan.
3. Hasil terbaik yang dapat digunakan sebagai lokasi mahasiswa ketika ketiga metode memiliki kode ruang kelas yang sama.
4. Jarak yang didapat dari hasil klasifikasi memiliki rata-rata 10 meter dari lokasi sebenarnya.
5. Selisih jarak dari hasil klasifikasi ketiga metode tidak lebih dari 5 meter.

6.2 Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Pengembangan terhadap pengambilan data lokasi mahasiswa saat itu dapat dikembangkan dengan

membuat *server* yang memiliki koneksi dengan *GPS* pada *smartphone* dan *server* tersebut terletak pada lantai tertinggi dari gedung.

2. Perhitungan dilakukan dengan kombinasi letak mahasiswa dalam ruang kelas dengan menjumlahkan jarak data lokasi oleh *server* (saran nomor 1) dengan *access point* (*Wi-Fi*).

3. Dapat menambahkan penggunaan sensor, seperti sensor cahaya dan sensor gerak untuk mengetahui keberadaan mahasiswa dalam ruang kelas atau penggunaan *bluetooth*.

Daftar Pustaka

[1] "Apa Itu Mobile Application?" [Daring]. Tersedia pada: http://cloudindonesia.com/apa-itu-*mobile*-application/. [Diakses: 08-Jul-2017].

[2] "4 sistem operasi smartphone paling dikenal," *For Guides*, 25-Feb-2016. .

[3] M. I. S.-U. Surabaya, "Android: Sistem Operasi Pada Smartphone | Universitas Surabaya (UBAYA)," *Universitas Surabaya (Ubaya)*. [Daring]. Tersedia pada: http://www.ubaya.ac.id/2014/content/articles_detail/7/Android--Sistem-Operasi-pada-Smartphone.html. [Diakses: 08-Jul-2017].

[4] "Global positioning system (GPS): Overview," *SpringerReference*.

[5] J. W. Betz, "Navstar Global Positioning System," in *Engineering Satellite-Based Navigation and Timing:Global Navigation Satellite Systems, Signals, and Receivers*, Wiley-IEEE Press, 2016, hal. 672-.

[6] "2013-1-00621-IF Bab2001.pdf.", *Binus University*.

[7] "Jurnal/Hasil Penelitian Kementerian Perindustrian." [Daring]. Tersedia pada: http://www.kemenperin.go.id/kajian. [Diakses: 15-Jul-2017].

[8] Y. Liu, J. Yang, dan M. Liu, "Recognition of QR Code with *mobile* phones," in *2008 Chinese Control and Decision Conference*, 2008, hal. 203–206.

[9] M. A. Banjarsari, I. Budiman, dan A. Farmadi, "PENERAPAN K-OPTIMAL PADA ALGORITMA KNN UNTUK PREDIKSI KELULUSAN TEPAT WAKTU MAHASISWA PROGRAM STUDI ILMU KOMPUTER FMIPA UNLAM BERDASARKAN IP SAMPAI DENGAN SEMESTER 4," *KLIK - Kumpul. J. ILMU Komput.*, vol. 2, no. 2, hal. 159–173, Apr 2016.

[10] A. R. Alfarisi, H. Tjandrasa, dan I. Arieshanti, "Perbandingan Performa antara Imputasi Metode Konvensional dan Imputasi dengan Algoritma Mutual Nearest Neighbor," *J. Tek. ITS*, vol. 2, no. 1, hal. A73–A76, Mar 2013.
[11] B. Santosa, "Data mining teknik pemanfaatan data untuk keperluan bisnis," *Yogyak. Graha Ilmu*, vol. 978, no. 979, hal. 756, 2007.
[12] "knn_references.pdf.", *Rumus Statistik* .
[13] G. H. Pradipto, "DASAR TEORI."
[14] "Kelebihan dan Kekurangan Rata-rata, Median dan Modus," *Rumus Statistik*. .
[15] "regresi_linier.pdf.", *Rumus Statistik* .
[16] "2011-1-00725-stif 2.pdf.", *Kemenperin* .
[17] ilham efendi, "Apa Yang di Maksud Dengan Server ?," *IT-Jurnal.com*, 23-Agu-2015. .
[18] T. M. Connolly dan C. E. Begg, *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.
[19] J. W. Satzinger, R. B. Jackson, dan S. D. Burd, *Systems Analysis and Design in a Changing World*. Cengage Learning, 2011.
[20] "JSON." [Daring]. Tersedia pada: http://www.json.org/json-id.html. [Diakses: 08-Jul-2017].

KODE SUMBER

Pada lampiran berikut dijelaskan mengenai kode sumber yang digunakan. Kode sumber menggunakan Bahasa Pemrograman C++. Berikut kode sumber yang dicantumkan:

1. Kode Sumber 1 - Decode Base64 merupakan kode sumber untuk mendekripsi hasil dari *QR Scanner* yang berisi *string* Base64.
2. Kode Sumber 2 - Fungsi md5Encode merupakan kode sumber untuk membanding hasil dari *QR Scanner* yang berisi *string* MD5 dengan data ruang kelas.
3. Kode Sumber 3 – Kelas MainPerkuliahanFragment merupakan kode sumber untuk *fragment* proses pembuatan API saat adanya trarnsaksi perpindahan *layout* atau pertama kali saat aplikasi dijalankan dan sebagai penanda proses verifikasi kehadiran (tandatangan atau wajah) dapat dilakukan.
4. Kode Sumber 4 – Kelas GoogleAPITracker merupakan kode sumber untuk mengambil data lokasi saat itu dengan memanfaatkan Google API sebagai *framework* GPS, lalu mengolah data tersebut dengan menggunakan ketiga metode yang selanjutnya akan digunakan untuk proses validasi kehadiran mahasiswa.
5. Kode Sumber 5 – Kelas LocationService merupakan kode sumber yang digunakan untuk mengatur agar proses pengambilan data lokasi, dapat berjalan secara *background process*, sehingga tidak akan mendapatkan interupsi dari fitur lain.

Kode Sumber 1 - Decode Base64

```
byte[] digest = Base64.decode(barcode.displayValue,
 Base64.DEFAULT);
String text = new String(digest, "UTF-8");
```

Kode Sumber 2 - Fungsi md5Encode

```
private String md5Encode(String value) {
    try {
        MessageDigest md = MessageDigest.getInstanc
e("MD5");
        md.update(value.getBytes());
        byte messageDigest[] = md.digest();

        StringBuilder hexString = new StringBuilder
();
        for (byte aMessageDigest : messageDigest) {

            String h = Integer.toHexString(0xFF & a
MessageDigest);
            while (h.length() < 2)
                h = "0" + h;
            hexString.append(h);
        }
        return hexString.toString();
    }
    catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return "";
}
```

Kode Sumber 3 – Kelas MainPerkuliahanFragment

```
package id.ac.its.sikemastc.activity.mahasiswa;

import android.app.Activity;
import android.app.Service;
```

```
5.  import android.content.Context;
6.  import android.content.Intent;
7.  import android.os.Handler;
8.  import android.os.IBinder;
9.  import android.os.StrictMode;
10. import android.support.annotation.Nullable;
11. import android.support.constraint.ConstraintLayout;

12. import android.support.v4.app.Fragment;
13. import android.os.Bundle;
14. import android.support.v4.content.ContextCompat;
15. import android.support.v4.widget.SwipeRefreshLayout
    ;
16. import android.support.v7.widget.LinearLayoutManage
    r;
17. import android.support.v7.widget.RecyclerView;
18. import android.util.Log;
19. import android.view.LayoutInflater;
20. import android.view.View;
21. import android.view.ViewGroup;
22. import android.widget.ImageButton;
23. import android.widget.ProgressBar;
24. import android.widget.TextView;
25. import android.widget.Toast;
26.
27. import com.android.volley.AuthFailureError;
28. import com.android.volley.Request;
29. import com.android.volley.Response;
30. import com.android.volley.VolleyError;
31. import com.android.volley.toolbox.StringRequest;
32.
33. import org.json.JSONArray;
34. import org.json.JSONException;
35. import org.json.JSONObject;
36.
37. import java.util.ArrayList;
38. import java.util.HashMap;
39. import java.util.List;
40. import java.util.Map;
41. import java.util.Objects;
42.
43. import id.ac.its.sikemastc.R;
```

```
44. import id.ac.its.sikemastc.activity.verifikasi_lokasi.GoogleAPITracker;
45. import id.ac.its.sikemastc.activity.verifikasi_lokasi.LocationService;
46. import id.ac.its.sikemastc.activity.verifikasi_tandatangan.MenuVerifikasiTandaTangan;
47. import id.ac.its.sikemastc.activity.verifikasi_wajah.VerifikasiWajahMenuActivity;
48. import id.ac.its.sikemastc.adapter.PerkuliahanAktifAdapter;
49. import id.ac.its.sikemastc.data.SikemasSessionManager;
50. import id.ac.its.sikemastc.model.Perkuliahan;
51. import id.ac.its.sikemastc.utilities.NetworkUtils;

52. import id.ac.its.sikemastc.utilities.SikemasDateUtils;
53. import id.ac.its.sikemastc.utilities.VolleySingleton;
54.
55. public class MainPerkuliahanFragment extends Fragment implements
56.         PerkuliahanAktifAdapter.PerkuliahanAktifOnClickHandler {
57.
58.     private final String TAG = MainPerkuliahanFragment.class.getSimpleName();
59.
60.     private TextView currentDate;
61. //    private TextView searchLoading;
62.     private SwipeRefreshLayout mSwipeRefreshLayout;

63.     private ProgressBar mLoadingIndicator;
64. //    private ProgressBar mSearchingIndicator;
65.     private RecyclerView mRecyclerView;
66.     private ConstraintLayout emptyView;
67.     private PerkuliahanAktifAdapter mPerkuliahanAktifAdapter;
68.     private String bundleIdUser;
69.     private String bundleNamaUser;
70.     private List<Perkuliahan> perkuliahanAktifMahasiswaList;
71.     private SikemasSessionManager session;
```

```
72.
73.     //Verifikasi Lokasi
74.     private ImageButton searchLocationButton;
75.     public static TextView location;
76.     public static ProgressBar mSearchingIndicator;

77.     public static TextView searchLoading;
78.
79.     @Override
80.     public View onCreateView(LayoutInflater inflate
    r, ViewGroup container,
81.                              Bundle savedInstanceSt
    ate) {
82.         final View view = inflater.inflate(R.layout
    .fragment_main_perkuliahan, container, false);
83.
84.         Bundle bundle = getArguments();
85.         if (bundle != null) {
86.             bundleIdUser = bundle.getString("id_mah
    asiswa");
87.             bundleNamaUser = bundle.getString("nama
    _mahasiswa");
88.             Log.d("bundleIdUser", bundleIdUser);
89.             Log.d("bundleNamaUser", bundleNamaUser)
    ;
90.         }
91.
92.         mSwipeRefreshLayout = (SwipeRefreshLayout)
    view.findViewById(R.id.swipe_refresh);
93.         mRecyclerView = (RecyclerView) view.findVie
    wById(R.id.rv_kelas_aktif);
94.         mLoadingIndicator = (ProgressBar) view.find
    ViewById(R.id.pb_loading_indicator);
95.         mSearchingIndicator = (ProgressBar) view.fi
    ndViewById(R.id.pb_searching_location);
96.         emptyView = (ConstraintLayout) view.findVie
    wById(R.id.empty_view);
97.         currentDate = (TextView) view.findViewById(
    R.id.tv_tanggal_hari_ini);
98.         searchLocationButton = (ImageButton) view.f
    indViewById(R.id.ib_refresh_location);
99.         searchLoading = (TextView) view.findViewByI
    d(R.id.tv_mencari_lokasi);
```

```
100.                location = (TextView) view.findViewB
    yId(R.id.tv_classroom_position);
101.
102.                currentDate = (TextView) view.findVi
    ewById(R.id.tv_tanggal_hari_ini);
103.                currentDate.setText(SikemasDateUtils
    .getCurrentDate(getActivity()));
104.
105.                mSwipeRefreshLayout.setColorSchemeCo
    lors(
106.                        ContextCompat.getColor(getCo
    ntext(), R.color.swipe_color_1),
107.                        ContextCompat.getColor(getCo
    ntext(), R.color.swipe_color_2),
108.                        ContextCompat.getColor(getCo
    ntext(), R.color.swipe_color_3),
109.                        ContextCompat.getColor(getCo
    ntext(), R.color.swipe_color_4));
110.
111.                LinearLayoutManager layoutManager =

112.                        new LinearLayoutManager(getA
    ctivity(), LinearLayoutManager.VERTICAL, false);
113.                mRecyclerView.setLayoutManager(layou
    tManager);
114.                mRecyclerView.setHasFixedSize(true);

115.
116.                return view;
117.            }
118.
119.            @Override
120.            public void onViewCreated(final View vie
    w, @Nullable Bundle savedInstanceState) {
121.                super.onViewCreated(view, savedInsta
    nceState);
122.
123.                if (!mSwipeRefreshLayout.isRefreshin
    g()) {
124.                    showLoading();
125.                }
126.
```

```
127.                if (GoogleAPITracker.finalResult[0]
    != "0") {
128.                    showLocationFounded();
129.                }
130.
131.                perkuliahanAktifMahasiswaList = new
    ArrayList<>();
132.                getPerkuliahanAktifList(bundleIdUser
    );
133.                mPerkuliahanAktifAdapter = new Perku
    liahanAktifAdapter(getActivity(), perkuliahanAktifM
    ahasiswaList, this);
134.                mRecyclerView.setAdapter(mPerkuliaha
    nAktifAdapter);
135.
136.                // Verifikasi Lokasi
137.                StrictMode.setThreadPolicy(new Stric
    tMode.ThreadPolicy.Builder().permitNetwork().build(
    ));
138.                final Intent intent = new Intent(vie
    w.getContext(), LocationService.class);
139.                intent.putExtra("NRP", this.bundleId
    User);
140.
141.                if (perkuliahanAktifMahasiswaList !=
     null) {
142.                    showPerkuliahanAktifDataView();

143.                } else {
144.                    showEmptyView();
145.                }
146.
147.                searchLocationButton.setOnClickListe
    ner(new View.OnClickListener() {
148.                    @Override
149.                    public void onClick(View arg0) {

150.                        getContext().startService(in
    tent);
151.                        showSearchingLocation();
152.                    }
153.                });
154.
```

```
155.                mSwipeRefreshLayout.setOnRefreshList
    ener(new SwipeRefreshLayout.OnRefreshListener() {
156.                    @Override
157.                    public void onRefresh() {
158.                        initiateRefresh();
159.                    }
160.                });
161.            }
162.
163.            @Override
164.            public void onClick(int buttonId, String
     idPerkuliahan, String kodeRuangan) {
165.                switch (buttonId) {
166.                    case R.id.btn_verifikasi_tandata
    ngan:
167.                        if (Objects.equals(GoogleAPI
    Tracker.finalResult[0], kodeRuangan)) {
168.                            Intent intentToVerifikas
    iTandaTangan = new Intent(getActivity(), MenuVerifi
    kasiTandaTangan.class);
169.                            intentToVerifikasiTandaT
    angan.putExtra("id_perkuliahan", idPerkuliahan);
170.                            intentToVerifikasiTandaT
    angan.putExtra("nrp_mahasiswa", bundleIdUser);
171.                            intentToVerifikasiTandaT
    angan.putExtra("nama_mahasiswa", bundleNamaUser);
172.                            startActivity(intentToVe
    rifikasiTandaTangan);
173.                        } else {
174.                            Toast.makeText(getContex
    t(), "Pastikan Anda berada pada ruangan yang benar
    atau " +
175.                                    "lakukan pencari
    an lokasi ulang", Toast.LENGTH_SHORT).show();
176.                            Log.d("Cek", GoogleAPITr
    acker.finalResult[0] + " " + kodeRuangan);
177.                        }
178.                        break;
179.
180.                    case R.id.btn_verifikasi_wajah:

181.                        if (Objects.equals(GoogleAPI
    Tracker.finalResult[0], kodeRuangan)) {
```

```
182.                            Intent intentToVerifikas
    iWajah = new Intent(getActivity(), VerifikasiWajahM
    enuActivity.class);
183.                            intentToVerifikasiWajah.
    putExtra("id_perkuliahan", idPerkuliahan);
184.                            intentToVerifikasiWajah.
    putExtra("nrp_mahasiswa", bundleIdUser);
185.                            intentToVerifikasiWajah.
    putExtra("nama_mahasiswa", bundleNamaUser);
186.                            startActivity(intentToVe
    rifikasiWajah);
187.                        } else {
188.                            Toast.makeText(getContex
    t(), "Pastikan Anda berada pada ruangan yang benar
    atau " +
189.                                                    "lakukan pencari
    an lokasi ulang", Toast.LENGTH_SHORT).show();
190.                        }
191.                        break;
192.                }
193.            }
194.
195.            private void showSearchingLocation() {
196.                location.setVisibility(View.GONE);
197.                mSearchingIndicator.setVisibility(Vi
    ew.VISIBLE);
198.                searchLoading.setVisibility(View.VIS
    IBLE);
199.            }
200.
201.            public static void showLocationFounded()
     {
202.                mSearchingIndicator.setVisibility(Vi
    ew.GONE);
203.                searchLoading.setVisibility(View.GON
    E);
204.                location.setVisibility(View.VISIBLE)
    ;
205.            }
206.
207.            private void showPerkuliahanAktifDataVie
    w() {
```

```
208.                mLoadingIndicator.setVisibility(View
    .GONE);
209.                emptyView.setVisibility(View.GONE);

210.                mRecyclerView.setVisibility(View.VIS
    IBLE);
211.            }
212.
213.            private void showLoading() {
214.                mRecyclerView.setVisibility(View.GON
    E);
215.                emptyView.setVisibility(View.GONE);

216.                mLoadingIndicator.setVisibility(View
    .VISIBLE);
217.            }
218.
219.            private void showEmptyView() {
220.                mRecyclerView.setVisibility(View.GON
    E);
221.                mLoadingIndicator.setVisibility(View
    .GONE);
222.                emptyView.setVisibility(View.VISIBLE
    );
223.            }
224.
225.            private void initiateRefresh() {
226.                Log.d(TAG, "initiate Refresh");
227.                getPerkuliahanAktifList(bundleIdUser
    );
228.            }
229.
230.            public void getPerkuliahanAktifList(fina
    l String idMahasiswa) {
231.                Log.d("idMahasiswa", idMahasiswa);
232.                if (!mSwipeRefreshLayout.isRefreshin
    g())
233.                    showLoading();
234.                StringRequest stringRequest = new St
    ringRequest(Request.Method.POST,
235.                        NetworkUtils.LIST_KELAS_MAHA
    SISWA_AKTIF,
```

```
236.                         new Response.Listener<String
    >() {
237.                             @Override
238.                             public void onResponse(S
    tring response) {
239.                                 Log.d(TAG, "Peserta
    Kehadiran Response: " + response);
240.                                 try {
241.                                     mPerkuliahanAkti
    fAdapter.clear();
242.
243.                                     JSONObject jsonO
    bject = new JSONObject(response);
244.                                     JSONArray listKe
    lasAktif = jsonObject.getJSONArray("listkelas");
245.                                     Log.d("length li
    stkelas", String.valueOf(listKelasAktif.length()));

246.
247.                                     for (int i = 0;
    i < listKelasAktif.length(); i++) {
248.                                         JSONObject d
    etailKelasAktif = listKelasAktif.getJSONObject(i);

249.                                         String idPer
    kuliahan = detailKelasAktif.getString("id");
250.                                         String perte
    muanKe = detailKelasAktif.getString("pertemuan");
251.                                         String statu
    sPerkuliahan = detailKelasAktif.getString("status_p
    erkuliahan");
252.                                         String statu
    sDosen = detailKelasAktif.getString("status_dosen")
    ;
253.                                         String hari
    = detailKelasAktif.getString("hari");
254.                                         String waktu
    Mulai = SikemasDateUtils.formatTime(detailKelasAkti
    f.getString("mulai"));
255.                                         String waktu
    Selesai = SikemasDateUtils.formatTime(detailKelasAk
    tif.getString("selesai"));
256.
```

```
257.                                        JSONObject k
    elas = detailKelasAktif.getJSONObject("kelas");
258.                                        String kodeR
    uangan = kelas.getString("kode_ruangan");
259.                                        String kodeS
    emester = kelas.getString("kode_semester");
260.                                        String kodeM
    k = kelas.getString("kode_matakuliah");
261.                                        String kelas
    Mk = kelas.getString("kode_kelas");
262.                                        String namaM
    k = kelas.getString("nama_kelas");
263.                                        String ruang
    MK = kelas.getString("nama_ruangan");
264.
265.                                        Perkuliahan
    perkuliahanAktifMahasiswa = new Perkuliahan(
266.                                                idPe
    rkuliahan, kodeRuangan, kodeMk, kodeSemester, namaM
    k,
267.                                                kela
    sMk, ruangMK, pertemuanKe, hari, waktuMulai,
268.                                                wakt
    uSelesai, statusDosen, statusPerkuliahan);
269.                                        perkuliahanA
    ktifMahasiswaList.add(perkuliahanAktifMahasiswa);
270.                                    }
271.
272.                                    if (mSwipeRefres
    hLayout.isRefreshing()) {
273.                                        mSwipeRefres
    hLayout.setRefreshing(false);
274.                                    }
275.
276.                                    if (listKelasAkt
    if.length() == 0) {
277.                                        mPerkuliahan
    AktifAdapter.notifyDataSetChanged();
278.                                        showEmptyVie
    w();
279.                                        Toast.makeTe
    xt(getActivity(), "Tidak ada Perkuliahan aktif yang
     dimuat",
```

```
                                                    Toast.LENGTH_SHORT).show();
                                        } else {
                                            mPerkuliahanAktifAdapter.notifyDataSetChanged();
                                            showPerkuliahanAktifDataView();
                                            Toast.makeText(getActivity(), "Berhasil Memuat Perkuliahan Aktif",
                                                    Toast.LENGTH_SHORT).show();
                                        }
                                    } catch (JSONException e) {
                                        e.printStackTrace();
                                        Toast.makeText(getActivity(), "Json error: " + e.getMessage(), Toast.LENGTH_LONG).show();
                                    }
                                }
                            }, new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {
                            Log.e(TAG, "Error: " + error.getMessage());
                            Toast.makeText(getActivity(), error.getMessage(), Toast.LENGTH_LONG).show();
                        }
                    }) {
                        @Override
                        protected Map<String, String> getParams() throws AuthFailureError {
                            // Posting parameters to login url
                            Map<String, String> params = new HashMap<String, String>();
                            params.put("nrp", idMahasiswa);
                            return params;
```

```
                    }
                };
                VolleySingleton.getmInstance(getActi
    vity()).addToRequestQueue(stringRequest);
            }
        }
```

Kode Sumber 4 – Kelas GoogleAPITracker

```
package id.ac.its.sikemastc.activity.verifikasi_lok
    asi;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.StrictMode;
import android.support.annotation.Nullable;
import android.util.Log;
import android.widget.Toast;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;

import com.google.android.gms.common.ConnectionResu
    lt;
import com.google.android.gms.common.GooglePlayServ
    icesUtil;
```

```
29. import com.google.android.gms.common.api.GoogleApiC
    lient;
30. import com.google.android.gms.location.LocationList
    ener;
31. import com.google.android.gms.location.LocationRequ
    est;
32. import com.google.android.gms.location.LocationServ
    ices;
33.
34. import com.opencsv.CSVWriter;
35. import org.json.JSONArray;
36. import id.ac.its.sikemastc.activity.mahasiswa.MainP
    erkuliahanFragment;
37.
38.
39. /**
40.  * Created by nurro on 3/3/2017.
41.  */
42.
43. public class GoogleAPITracker extends Service imple
    ments LocationListener {
44.     //Variable for Context
45.     private Context context;
46.
47.     //GoogleAPI Client
48.     private GoogleApiClient googleApiClient;
49.
50.     //Variable for Attribute GPS
51.     private String providerName;
52.     private LocationRequest locationRequest;
53.     private double latitude;
54.     private double longitude;
55.     private double accuracy;
56.     private double altitude;
57.
58.     //Variable for Method Result
59.     private ArrayList<String[]> resultKNN;
60.     private ArrayList<String[]> result
    Average;
61.     private ArrayList<String[]> resultRegression;
62.     public static String[] finalResult = new String
    []{"0", "0", "0", "0"};
63.
```

```
64.     // The minimum distance to change Updates in me
    ters
65.     private static final long minDistance = 0; // 1
    0 meters
66.
67.     // The minimum time between updates in millisec
    onds
68.     private static final long minTime = 1000 * 1 *
    1; // 1 menit
69.
70.     // Max iteration
71.     private int iteration = 0;
72.
73.     // Connection to SERVER
74.     private String link = "http://10.151.31.201/sik
    emas/public/get_ruang";
75.     private String[][] dataPlaces;
76.     private InputStream inputStream = null;
77.     private String line;
78.     private String result;
79.     private int dataLength;
80.     private ArrayList<String[]> dataIteration;
81.     private String NRP;
82.
83.     private void setContext(Context context) {
84.         this.context = context;
85.     }
86.
87.     public GoogleAPITracker(final Context context2,
     String NRP) {
88.         setContext(context2);
89.         this.NRP = NRP;
90.     }
91.
92.     private boolean checkPlayServices() {
93.         int result = GooglePlayServicesUtil.isGoogl
    ePlayServicesAvailable(this.context);
94.         if (result != ConnectionResult.SUCCESS) {
95.             if (GooglePlayServicesUtil.isUserRecove
    rableError(result)) {
96.                 Toast.makeText(this.context,
```

```
97.                         "This device is supported.
    Please download Google Play Services", Toast.LENGTH
    _LONG)
98.                         .show();
99.             }
100.                    else {
101.                        Toast.makeText(this.context,
                                
102.                                "This device is not
    supported", Toast.LENGTH_LONG)
103.                                .show();
104.                    }
105.                    return false;
106.                }
107.                return true;
108.            }
109.
110.            public boolean isGoogleApiClientConnecte
    d() {
111.                return googleApiClient != null && go
    ogleApiClient.isConnected();
112.            }
113.
114.            private void startAPI() {
115.                if (googleApiClient == null) {
116.                    googleApiClient = new GoogleApiC
    lient.Builder(this.context).addApi(LocationServices
    .API)
117.                            .addConnectionCallbacks(
    new GoogleApiClient.ConnectionCallbacks(){
118.                                @Override
119.                                public void onConnec
    tionSuspended(int cause) {
120.
121.                                }
122.
123.                                @Override
124.                                public void onConnec
    ted(Bundle connectionHint) {
125.
126.                                }
127.
```

```
128.                            }).addOnConnectionFailed
    Listener(new GoogleApiClient.OnConnectionFailedList
    ener() {
129.                                @Override
130.                                public void onConnec
    tionFailed(ConnectionResult result) {
131.                                }
132.
133.                            }).build();
134.                    googleApiClient.connect();
135.                }
136.                else {
137.                    googleApiClient.connect();
138.                }
139.            }
140.
141.            private void requestUpdateLocation(final
     LocationListener listener) {
142.                locationRequest = LocationRequest.cr
    eate();
143.                locationRequest.setPriority(Location
    Request.PRIORITY_HIGH_ACCURACY);
144.                locationRequest.setInterval(minTime)
    ;
145.
146.                new Handler().postDelayed(new Runnab
    le() {
147.                    @Override
148.                    public void run() {
149.                        try {
150.                            LocationServices.FusedLo
    cationApi.requestLocationUpdates(googleApiClient, l
    ocationRequest, listener);
151.                        }
152.                        catch (SecurityException se)
     {
153.                            Log.d("Security Exceptio
    n", se.toString());
154.                            se.printStackTrace();
155.                        }
156.                        catch (Exception ex){
157.                            Log.d("Ex Request", ex.t
    oString());
```

```
158.                             ex.printStackTrace();
159.                         }
160.                     }
161.                 }, minTime);
162.             }
163.
164.
165.             public void getLocation() {
166.                 try {
167.                     if (checkPlayServices()) {
168.                         Toast.makeText(this.context,
    "Pencarian Dimulai", Toast.LENGTH_SHORT).show();
169.                         Log.d("Pencarian Lokasi", "P
    encarian Lokasi Dimulai");
170.                         this.iteration = 0;
171.                         this.dataIteration = new Arr
    ayList<>();
172.                         getPlaces();
173.                         startAPI();
174.                         requestUpdateLocation(this);

175.                     }
176.                 }
177.                 catch (Exception ex) {
178.                     Log.d("Error Google API", ex.toS
    tring());
179.                 }
180.             }
181.
182.
183.             //Get Latitude Value
184.             public double getLatitude() {
185.                 return latitude;
186.             }
187.
188.             //Get Longitude Value
189.             public double getLongitude() {
190.                 return longitude;
191.             }
192.
193.             //Get Altitude Value
194.             public double getAltitude() {
195.                 return altitude;
```

```
196.            }
197.
198.            //Get Accuracy Value
199.            public double getAccuracy() {
200.                return accuracy;
201.            }
202.
203.            public void stopUsingAPI() {
204.                Toast.makeText(this.context, "Pencar
    ian Selesai", Toast.LENGTH_SHORT).show();
205.                if (googleApiClient != null) {
206.                    googleApiClient.disconnect();
207.                }
208.            }
209.
210.            private void getPlaces() {
211.                StrictMode.setThreadPolicy(new Stric
    tMode.ThreadPolicy.Builder().permitNetwork().build(
    ));
212.                line = null;
213.                try {
214.                    URLConnection url = new URL(link
    ).openConnection();
215.
216.                    HttpURLConnection con = (HttpURL
    Connection)url;
217.
218.                    inputStream = new BufferedInputS
    tream(con.getInputStream());
219.
220.                    BufferedReader br = new Buffered
    Reader(new InputStreamReader(inputStream));
221.                    StringBuilder sb = new StringBui
    lder();
222.
223.                    while((line = br.readLine()) !=
    null){
224.                        sb.append(line+"\n");
225.                    }
226.                    inputStream.close();
227.                    result = sb.toString();
228.
229.                }
```

```
230.                catch (Exception ex){
231.                    ex.printStackTrace();
232.                }
233.
234.                try {
235.                    JSONArray jsonArray = new JSONAr
    ray(result);
236.                    dataLength = jsonArray.length();

237.                    dataPlaces = new String[jsonArra
    y.length()][7];
238.                    int j=0;
239.                    for(int i = 0; i < this.dataLeng
    th; i++){
240.                        if (jsonArray.getJSONObject(
    i).getString("longitude") != "0" || jsonArray.getJS
    ONObject(i).getString("latitude") != "0") {
241.                            dataPlaces[j][0] = jsonA
    rray.getJSONObject(i).getString("id");
242.                            dataPlaces[j][1] = jsonA
    rray.getJSONObject(i).getString("nama");
243.                            dataPlaces[j][2] = jsonA
    rray.getJSONObject(i).getString("latitude");
244.                            dataPlaces[j][3] = jsonA
    rray.getJSONObject(i).getString("longitude");
245.                            dataPlaces[j][4] = jsonA
    rray.getJSONObject(i).getString("altitude");
246.                            j++;
247.                        }
248.                    }
249.                    this.dataLength = j;
250.
251.                }
252.
253.                catch (Exception ex){
254.                    ex.printStackTrace();
255.                }
256.            }
257.
258.            private void KNN() {
259.                double tempDistance, tempAltitude;
260.                String result[][] = new String[this.
    dataLength][4];
```

```
261.                this.resultKNN = new ArrayList<>();
262.                int counter = 0;
263.                String[] tempData = new String[3];
264.                Location loc1, loc2;
265.
266.                for (int i = 0; i < this.iteration;
    i++) {
267.                    loc1 = new Location("");
268.                    loc1.setLongitude(Double.parseDo
    uble(this.dataIteration.get(i)[1]));
269.                    loc1.setLatitude(Double.parseDou
    ble(this.dataIteration.get(i)[2]));
270.                    for (int j = 0; j < this.dataLen
    gth; j++) {
271.                        loc2 = new Location("");
272.                        loc2.setLatitude(Double.pars
    eDouble(this.dataPlaces[j][2]));
273.                        loc2.setLongitude(Double.par
    seDouble(this.dataPlaces[j][3]));
274.                        tempDistance = loc1.distance
    To(loc2);
275.                        if (tempData[0] == null) {
276.                            tempData[0] = Double.toS
    tring(tempDistance);
277.                            tempData[1] = this.dataP
    laces[j][0];
278.                            tempData[2] = this.dataP
    laces[j][1];
279.                        }
280.                        else if (tempData[0] != null
    ) {
281.                            if (tempDistance < Doubl
    e.parseDouble(tempData[0])) {
282.                                tempData[0] = Double
    .toString(tempDistance);
283.                                tempData[1] = this.d
    ataPlaces[j][0];
284.                                tempData[2] = this.d
    ataPlaces[j][1];
285.                            }
286.                        }
287.                        else {
```

```
288.                            tempData[0] = Integer.to
    String(0);
289.                            tempData[1] = Integer.to
    String(0);
290.                            tempData[2] = Integer.to
    String(0);
291.                        }
292.                    }
293.
294.                    boolean available = false;
295.                    for (int j = 0; j < counter; j++
    ) {
296.                        if (result[j][0].equals(temp
    Data[1])) {
297.                            int tempCounter = Intege
    r.parseInt(result[j][3])+1;
298.                            result[j][3] = Integer.t
    oString(tempCounter);
299.                            available = true;
300.                            Log.d("Result " + Intege
    r.toString(i + 1), result[j][1] + " n: " + result[j
    ][3]);
301.                            Log.d("Data Iteration "
    + Integer.toString(i + 1), tempData[1]);
302.                            break;
303.                        }
304.                    }
305.                    if (!available) {
306.                        result[counter][0] = tempDat
    a[1];
307.                        result[counter][1] = tempDat
    a[2];
308.                        result[counter][2] = tempDat
    a[0];
309.                        result[counter][3] = Integer
    .toString(1);
310.                        Log.d("Result " + Integer.to
    String(i + 1), result[counter][1] + " n: " + result
    [counter][3]);
311.                        Log.d("Data Iteration " + In
    teger.toString(i + 1), tempData[1]);
312.                        counter++;
313.                    }
```

```
314.                }
315.                for (int i = 0; i < counter; i++) {

316.                    this.resultKNN.add(result[i]);
317.                }
318.                int temp1 = 0, temp2 = 0;
319.                int idResult = 0;
320.                double distance = Double.parseDouble
    (result[0][2]);
321.                temp1 = Integer.parseInt(result[0][3
    ]);
322.                idResult = Integer.parseInt(result[0
    ][0]);
323.                String placeResult = result[0][1];
324.                for (int i = 1; i < counter; i++) {

325.                    temp2 = Integer.parseInt(result[
    i][3]);
326.                    if (temp2 > temp1) {
327.                        temp1 = temp2;
328.                        distance = Double.parseDoubl
    e(result[i][2]);
329.                        idResult = Integer.parseInt(
    result[i][0]);
330.                        placeResult = result[i][1];

331.                    }
332.                }
333.                Log.d("ID: " + idResult, placeResult
    );
334.                this.resultKNN.add(new String[]{Inte
    ger.toString(idResult), placeResult, Double.toStrin
    g(distance), "Final"});
335.            }
336.
337.            private void xyMethod() {
338.                double Simple
    AverageLongitude = 0, Simple
    AverageLatitude = 0, Simple
    AverageAltitude = 0, tempDistance = 0;
339.                Location loc1, loc2;
340.                this.resultSimple
    Average = new ArrayList<>();
```

```
341.                String[][] result = new String[1][3]
    ;
342.
343.                for (int i = 0; i < this.iteration;
    i++) {
344.                    SimpleAverageLongitude += Double.parseDouble(this.dataIteration.get(i)[1]);
345.                    SimpleAverageLatitude += Double.parseDouble(this.dataIteration.get(i)[2]);
346.                    SimpleAverageAltitude += Double.parseDouble(this.dataIteration.get(i)[3]);
347.                }
348.                SimpleAverageLongitude /= this.iteration;
349.                SimpleAverageLatitude /= this.iteration;
350.                SimpleAverageAltitude /= this.iteration;
351.
352.                this.resultSimpleAverage.add(new String[]{Double.toString(SimpleAverageLongitude), Double.toString(SimpleAverageLatitude), Double.toString(SimpleAverageAltitude)});
353.
354.                Log.d("SimpleAverage XY", "Long: " + SimpleAverageLongitude + "\nLat: " + SimpleAverageLatitude + "\nAlt: " + SimpleAverageAltitude);
355.                loc1 = new Location("");
356.                loc1.setLatitude(SimpleAverageLatitude);
357.                loc1.setLongitude(SimpleAverageLongitude);
358.                for (int i = 0; i < this.dataLength;
     i++) {
359.                    loc2 = new Location("");
360.                    loc2.setLatitude(Double.parseDouble(this.dataPlaces[i][2]));
```

```
361.                     loc2.setLongitude(Double.parseDo
    uble(this.dataPlaces[i][3]));
362.                     Log.d("Check " + i, "ID: " + thi
    s.dataPlaces[i][0] + "\nPlace: " + this.dataPlaces[
    i][1] + "\nDistance: " +
363.                             loc1.distanceTo(loc2));

364.
365.                     this.resultSimple
    Average.add(new String[]{this.dataPlaces[i][0], thi
    s.dataPlaces[i][1], Double.toString(loc1.distanceTo
    (loc2))});
366.
367.                     if (tempDistance == 0 || tempDis
    tance > loc1.distanceTo(loc2)) {
368.                         tempDistance = loc1.distance
    To(loc2);
369.                         result[0][0] = this.dataPlac
    es[i][0];
370.                         result[0][1] = this.dataPlac
    es[i][1];
371.                         result[0][2] = Double.toStri
    ng(tempDistance);
372.                     }
373.                 }
374.                 this.resultSimple
    Average.add(new String[]{result[0][0], result[0][1]
    ,result[0][2]});
375.                 Log.d("ID: " + result[0][0], result[
    0][1]);
376.             }
377.
378.             private void linearRegression() {
379.                 Location loc1, loc2;
380.                 double sumX = 0, sumY = 0, sumX2 = 0
    , sumY2 = 0, sumXY = 0, medianX, medianY, a, b, res
    ultX, resultY, tempDistance = 0,
381.                         Simple
    AverageAltitude = 0;
382.                 this.resultRegression = new ArrayLis
    t<>();
383.                 String[][] result = new String[1][3]
    ;
```

```
384.
385.                 for (int i = 0; i < this.iteration;
    i++) {
386.                     sumX += Double.parseDouble(this.
    dataIteration.get(i)[1]);
387.                     sumY += Double.parseDouble(this.
    dataIteration.get(i)[2]);
388.                     sumX2 += (Double.parseDouble(thi
    s.dataIteration.get(i)[1])*Double.parseDouble(this.
    dataIteration.get(i)[1]));
389.                     sumY2 += (Double.parseDouble(thi
    s.dataIteration.get(i)[2])*Double.parseDouble(this.
    dataIteration.get(i)[2]));
390.                     sumXY += (Double.parseDouble(thi
    s.dataIteration.get(i)[1])*Double.parseDouble(this.
    dataIteration.get(i)[2]));
391.                     SimpleAverageAltitude += Double.parseDouble(this.dataIter
    ation.get(i)[3]);
392.                 }
393.                 SimpleAverageAltitude /= this.iteration;
394.                 int half = this.iteration/2;
395.                 if(this.iteration % 2 == 0) {
396.                     medianX = (Double.parseDouble(th
    is.dataIteration.get(half-
    1)[1]) + Double.parseDouble(this.dataIteration.get(
    half)[1]))/2;
397.                     medianY = (Double.parseDouble(th
    is.dataIteration.get(half-
    1)[2]) + Double.parseDouble(this.dataIteration.get(
    half)[2]))/2;
398.                 }
399.                 else {
400.                     medianX = Double.parseDouble(thi
    s.dataIteration.get(half)[1]);
401.                     medianY = Double.parseDouble(thi
    s.dataIteration.get(half)[2]);
402.                 }
403.                 b = ((this.iteration*sumXY) - (sumX*
    sumY))/((this.iteration*sumX2) - (sumX*sumX));
404.                 a = (sumY/this.iteration) - (b*(sumX
    /this.iteration));
```

```
405.                resultX = (medianY-a)/b;
406.                resultY = a + (b*medianX);
407.
408.                this.resultRegression.add(new String
    []{Double.toString(sumX), Double.toString(sumY), Do
    uble.toString(sumX2), Double.toString(sumY2),
409.                        Double.toString(sumXY), Doub
    le.toString(medianX), Double.toString(medianY), Dou
    ble.toString(a),
410.                        Double.toString(b), Double.t
    oString(resultX), Double.toString(resultY), Double.
    toString(Simple AverageAltitude)});
411.
412.                Log.d("Regresi Linier", "Sum X: " +
    sumX + "\nSum Y: " + sumY + "\nSum X2: " + sumX2 +
    "\nSum Y2: " + sumY2 +
413.                        "\nSum XY: " + sumXY + "\nMe
    dian X: " + medianX + "\nMedian Y: " + medianY + "\
    nA: " + a + "\nB: " + b +
414.                        "\nLong: " + resultX + "\nLa
    t: " + resultY);
415.
416.                loc1 = new Location("");
417.                loc1.setLongitude(resultX);
418.                loc1.setLatitude(resultY);
419.
420.                for (int i = 0; i < this.dataLength;
     i++) {
421.                    loc2 = new Location("");
422.                    loc2.setLatitude(Double.parseDou
    ble(this.dataPlaces[i][2]));
423.                    loc2.setLongitude(Double.parseDo
    uble(this.dataPlaces[i][3]));
424.                    Log.d("Check " + i, "ID: " + thi
    s.dataPlaces[i][0] + "\nPlace: " + this.dataPlaces[
    i][1] + "\nDistance: " +
425.                            loc1.distanceTo(loc2));

426.
427.                    this.resultRegression.add(new St
    ring[]{this.dataPlaces[i][0], this.dataPlaces[i][1]
    , Double.toString(loc1.distanceTo(loc2))});
428.
```

```
429.                    if (tempDistance == 0 || tempDis
    tance > loc1.distanceTo(loc2)) {
430.                        tempDistance = loc1.distance
    To(loc2);
431.                        result[0][0] = this.dataPlac
    es[i][0];
432.                        result[0][1] = this.dataPlac
    es[i][1];
433.                        result[0][2] = Double.toStri
    ng(tempDistance);
434.                    }
435.                }
436.                this.resultRegression.add(new String
    []{result[0][0], result[0][1],result[0][2]});
437.                Log.d("ID: " + result[0][0], result[
    0][1]);
438.            }
439.
440.            private void createCSVFile() {
441.                String baseDir = android.os.Environm
    ent.getExternalStorageDirectory() + "/Sikemas/lokas
    i/";
442.                Long ts = System.currentTimeMillis()
    /1000;
443.                String fileName = NRP + "-
    " + ts.toString() + ".csv";
444.                String filePath = baseDir + fileName
    ;
445.                File f = new File(baseDir);
446.                if(!f.exists())
447.                {
448.                    f.mkdirs();
449.                }
450.                CSVWriter writer;
451.                try {
452.                    if (this.iteration == 10) {
453.                        writer = new CSVWriter(new F
    ileWriter(filePath));
454.                        // Write Data Iteration
455.                        writer.writeNext(new String[
    ]{"Data Iterasi"});
```

```
456.                         writer.writeNext(new String[
    ]{"No", "Longitude", "Latitude", "Altitude", "Accur
    acy"});
457.                         for (int i = 0; i < this.ite
    ration; i++) {
458.                             writer.writeNext(new Str
    ing[]{Integer.toString(i+1), this.dataIteration.get
    (i)[1], this.dataIteration.get(i)[2],
459.                                                 this.dataIterati
    on.get(i)[3], this.dataIteration.get(i)[4]});
460.                         }
461.                         writer.writeNext(new String[
    ]{""});
462.                         writer.writeNext(new String[
    ]{""});
463.
464.                         // Write KNN Result
465.                         writer.writeNext(new String[
    ]{"KNN"});
466.                         writer.writeNext(new String[
    ]{"No", "ID", "Name", "Distance", "Counter"});
467.                         for (int i = 0; i < this.res
    ultKNN.size(); i++) {
468.                             writer.writeNext(new Str
    ing[]{Integer.toString(i+1), this.resultKNN.get(i)[
    0], this.resultKNN.get(i)[1],
469.                                                 this.resultKNN.g
    et(i)[2], this.resultKNN.get(i)[3]});
470.                         }
471.
472.                         writer.writeNext(new String[
    ]{""});
473.                         writer.writeNext(new String[
    ]{""});
474.
475.                         // Write Simple
    Average Result
476.                         writer.writeNext(new String[
    ]{"Simple Average"});
477.                         writer.writeNext(new String[
    ]{"No", "ID", "Name", "Distance", "", "Simple
    AverageLongitude", "Simple
    AverageLatitude", "Simple AverageAltitude"});
```

```
478.                         for (int i = 1; i < this.res
    ultSimple Average.size(); i++) {
479.                             writer.writeNext(new Str
    ing[]{Integer.toString(i), this.resultSimple
    Average.get(i)[0], this.resultSimple
    Average.get(i)[1],
480.                                     this.resultSimpl
    e Average.get(i)[2], "", this.resultSimple
    Average.get(0)[0], this.resultSimple
    Average.get(0)[1],
481.                                     this.resultSimpl
    e Average.get(0)[2]});
482.                         }
483.                         writer.writeNext(new String[
    ]{""});
484.                         writer.writeNext(new String[
    ]{""});
485.
486.                         // Write Regression
487.                         writer.writeNext(new String[
    ]{"Regression"});
488.                         writer.writeNext(new String[
    ]{"No", "ID", "Name", "Distance", "", "SumX", "SumY
    ", "SumX2", "SumY2", "SumXY",
489.                                         "MedianX", "MedianY"
    , "A", "B", "ResultX", "ResultY", "Simple
    AverageAltitude"});
490.                         for (int i = 1; i < this.res
    ultRegression.size(); i++) {
491.                             writer.writeNext(new Str
    ing[]{Integer.toString(i), this.resultRegression.ge
    t(i)[0], this.resultRegression.get(i)[1],
492.                                     this.resultRegre
    ssion.get(i)[2], "", this.resultRegression.get(0)[0
    ], this.resultRegression.get(0)[1],
493.                                     this.resultRegre
    ssion.get(0)[2], this.resultRegression.get(0)[3], t
    his.resultRegression.get(0)[4],
494.                                     this.resultRegre
    ssion.get(0)[5], this.resultRegression.get(0)[6], t
    his.resultRegression.get(0)[7],
```

```
495.                                 this.resultRegre
    ssion.get(0)[8], this.resultRegression.get(0)[9], t
    his.resultRegression.get(0)[10],
496.                                 this.resultRegre
    ssion.get(0)[11]});
497.                         }
498.                         writer.writeNext(new String[
    ]{""});
499.                         writer.writeNext(new String[
    ]{""});
500.
501.                         writer.close();
502.                     }
503.
504.                     String id = "0", place = "0";
505.                     double distance = 100000000;
506.
507.                     if (distance > Double.parseDoubl
    e(this.resultKNN.get(this.resultKNN.size()-
    1)[2])) {
508.                         if (Double.parseDouble(this.
    resultKNN.get(this.resultKNN.size()-1)[2]) > 0) {
509.                             id = this.resultKNN.get(
    this.resultKNN.size()-1)[0];
510.                             place = this.resultKNN.g
    et(this.resultKNN.size()-1)[1];
511.                             distance = Double.parseD
    ouble(this.resultKNN.get(this.resultKNN.size()-
    1)[2]);
512.                         }
513.                     }
514.                     if (distance > Double.parseDoubl
    e(this.result Simple  Average .get(this.result Simple
     Average .size()-1)[2])) {
515.                         if (Double.parseDouble(this.
    result Simple  Average .get(this.result Simple
     Average .size()-1)[2]) > 0) {
516.                             id = this.result Simple
     Average .get(this.result Simple  Average .size()-
    1)[0];
517.                             place = this.result Simpl
    e  Average .get(this.result Simple  Average .size()-
    1)[1];
```

```
518.                             distance = Double.parseD
    ouble(this.resultSimple
    Average.get(this.resultSimple Average.size()-
    1)[2]);
519.                         }
520.                     }
521.                     if (distance > Double.parseDoubl
    e(this.resultRegression.get(this.resultRegression.s
    ize()-1)[2])) {
522.                         if (Double.parseDouble(this.
    resultRegression.get(this.resultRegression.size()-
    1)[2]) > 0) {
523.                             id = this.resultRegressi
    on.get(this.resultRegression.size()-1)[0];
524.                             place = this.resultRegre
    ssion.get(this.resultRegression.size()-1)[1];
525.                             distance = Double.parseD
    ouble(this.resultRegression.get(this.resultRegressi
    on.size()-1)[2]);
526.                         }
527.                     }
528.                     Log.d("Final Result", place);
529.                     finalResult[0] = id;
530.                     finalResult[1] = place;
531.                     finalResult[2] = Double.toString
    (distance);
532.                     MainPerkuliahanFragment.location
    .setText(finalResult[1]);
533.                     MainPerkuliahanFragment.showLoca
    tionFounded();
534.                     //sendFinalResult();
535.
536.                 } catch (IOException e) {
537.                     e.printStackTrace();
538.                 }
539.             }
540.
541.             @Nullable
542.             @Override
543.             public IBinder onBind(Intent intent) {
544.                 return null;
545.             }
546.
```

```
547.            @Override
548.            public void onLocationChanged(Location l
    ocation) {
549.                this.accuracy = location.getAccuracy
    ();
550.                this.latitude = location.getLatitude
    ();
551.                this.longitude = location.getLongitu
    de();
552.                this.altitude = location.getAltitude
    ();
553.                if(this.accuracy <= 20) {
554.                    if (this.iteration > 0 && this.l
    ongitude != Double.parseDouble(this.dataIteration.g
    et(this.iteration-1)[1]) &&
555.                            this.latitude != Double.
    parseDouble(this.dataIteration.get(this.iteration-
    1)[2])) {
556.                        Log.d("Data", "Longitude: "
    + getLongitude() + "\nLatitude: " +
557.                                getLatitude() + "\nA
    ccuracy: " + getAccuracy() + "\nAltitude: " + getAl
    titude());
558.                        this.dataIteration.add(new S
    tring[]{Integer.toString(iteration + 1), Double.toS
    tring(getLongitude()),
559.                                Double.toString(getL
    atitude()), Double.toString(getAltitude()), Double.
    toString(getAccuracy())});
560.                        this.iteration++;
561.                    }
562.                    else if (this.iteration == 0) {

563.                        Log.d("Data", "Longitude: "
    + getLongitude() + "\nLatitude: " +
564.                                getLatitude() + "\nA
    ccuracy: " + getAccuracy() + "\nAltitude: " + getAl
    titude());
565.                        this.dataIteration.add(new S
    tring[]{Integer.toString(iteration + 1), Double.toS
    tring(getLongitude()),
```

```
566.                                 Double.toString(getL
    atitude()), Double.toString(getAltitude()), Double.
    toString(getAccuracy())});
567.                         this.iteration++;
568.                     }
569.                     else {
570.                         stopUsingAPI();
571.                     }
572.                     if (this.iteration > 0) {
573.                         try {
574.                             KNN();
575.                             xyMethod();
576.                             linearRegression();
577.                             createCSVFile();
578.                         }
579.                         catch (Exception ex) {
580.
581.                         }
582.                     }
583.                 }
584.             }
585.
586.         }
```

Kode Sumber 5 – Kelas LocationService

```
1.  package id.ac.its.sikemastc.activity.verifikasi_lok
    asi;
2.
3.  import android.app.Service;
4.  import android.content.Intent;
5.  import android.os.IBinder;
6.  import android.support.annotation.Nullable;
7.  import android.util.Log;
8.
9.  /**
10.  * Created by nurro on 6/2/2017.
11.  */
12.
13. public class LocationService extends Service {
14.     private GoogleAPITracker googleAPI;
15.
```

```
16.     @Nullable
17.     @Override
18.     public IBinder onBind(Intent intent) {
19.         return null;
20.     }
21.
22.     @Override
23.     public void onDestroy() {
24.         super.onDestroy();
25.         this.googleAPI.stopUsingAPI();
26.     }
27.
28.     @Override
29.     public int onStartCommand(Intent intent, int fl
    ags, int startId) {
30.         Log.d("Memulai Service Lokasi", intent.getS
    tringExtra("NRP"));
31.         googleAPI = new GoogleAPITracker(this, inte
    nt.getStringExtra("NRP"));
32.         if (!googleAPI.isGoogleApiClientConnected()
    ) {
33.             googleAPI.getLocation();
34.         }
35.         if(googleAPI == null)
36.             stopSelf();
37.         return Service.START_NOT_STICKY;
38.     }
39. }
```

BIODATA PENULIS

Adian Latifa Nurrohman dilahirkan di Tangerang pada tanggal 27 April 1995. Pendidikan penulis tempuh dengan SD Jaya Suti Abadi Tambun Selatan, SMPN 1 Purworejo, SMAN 1 Purworejo dan S1 Teknik Informatika ITS (2013-2017).
Selama kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika ITS (HMTC), diantaranya adalah sebagai Staf Departemen Hubungan Luar (2014-2015). Penulis juga aktif dalam kegiatan kepanitiaan Schematics dengan menjadi staf dana dan usaha pada Schematics 2014 dan 2015. Penulis juga merupakan anggota PSM dengan sebutan LA14. Selain aktif organisasi, dalam bidang akademik, penulis pernah menjadi asisten dosen pada matakuliah Manajemen Basis Data .

Kritik dan saran sangat diharapkan guna meningkatkan kualitas penulisan selanjutnya. Untuk itu, silahkan kirim kritik dan saran ke : adian273645@gmail.com